UNIVERSITY OF OSLO
Department of Informatics

Visualization of PRADS Output Data Using Open-source Visualization Tools For Improved Log Analysis

Dawit Hailu Desta
dawithd@ifi.uio.no

Network and System Administration

Oslo University College

May 20, 2014

Visualization of PRADS Output Data Using Open-source Visualization Tools For Improved Log Analysis

Dawit Hailu Desta
dawithd@ifi.uio.no

Network and System Administration
Oslo University College

May 20, 2014

Abstract

The ever growing network traffic complexity has brought new threats and vulnerabilities that can affect our day to day activities. This lead to high demand for network monitoring and detection system to tackle the emerging threats . Consequently the inspection and assessment of security incidents has become a daily activity for network and system administrators. Network analysts need to have the awareness about every network activity, the status of the network system and the network assets in the network system. Many tools have been developed to detect and monitor network activities using active scanning and passive scanning mechanisms. This thesis focuses on Passive Real-time Detection System, PRADS, the smartest and powerful asset detection system that creates the network awareness required by network analysts. This asset detection tool reports everything it detects in the network and puts it in a log file. Analyzing the log file using a traditional way of textual log analysis is really hard to analyze and inspect especially when the log file is big. So this thesis tries to implement the visualization of the PRADS log file using open-source visualization tools for better analysis and network awareness. A survey is made on the available open-source visualization tools with regards to their suitability and applicability to visualize PRADS output data. The second phase of the project continues by suggesting different visualization methods for PRADS data. Finally a prototype is developed to demonstrate a proof of concept by using the relevant open-source tools Afterglow and Graphviz. The prototype developed tries to visualize wide range of log file data collected by PRADS in different network scenarios that is often difficult to readily search for patterns and trends using traditional log file analysis methods. The data from PRADS is parsed and fed to the Afterglow scripts to produce inputs suitable for use by the graph layouts in Graphviz. This project tries to successfully show the importance of visualizing log files to reveal the most important properties such as the status of running services on the network, rapid recognition of patters and trends, creation of status awareness on the network mapping. The final results achieved show that the project is successful and the project paves a way for further similar researches.

Acknowledgements

I would like to express my appreciation to the following people for their support in many different ways:

- First of all I would like to thank my beloved family for all their help and understanding; they have been there whenever I need them.
- I am also grateful to Norway, University of Oslo, and Oslo and Akershus University College for providing me a conducive environment to proceed my study.
- Professor Hårek Haugerud who has been supervising me throughout the whole thesis and encouraging me during some challenging times. He has spent a lot of time reading my reports and guiding me. He was the one giving me confidence whenever I encounter challenges in my project.
- Kyrre Begnum for his useful seminars on how to write a good thesis, his excellent teaching methods and his helpful attitude to support students when needed.
- Redpill Linpro for proposing me this interesting project to be the master thesis I work on.
- Ismail Hassan for his positive attitude when asked for help and his interesting classes.
- Kacper Wysocki for providing me the necessary information on this project whenever I ask for clarification.
- Amir Maqbool Ahmed for helping me to create the experiment environment setup and his cooperation when I encounter problems with the lab-environment.
- AfterGlow developer Raffael Marty deserves a lot of credit for developing the main tool implemented in this project and for his inspiring ideas which made me to have the interest to implement it.
- Finally my dear friends especially Abraha Desta who have always been very helpful by commenting on my work and giving me a hand whenever I needed help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter the first section explains the motivation for choosing the visualization of the output from PRADS which is a passive detection system and its importance for network and system administrators. The problem statement and research questions are defined in the second section and finally the structure of the thesis report is presented in the last section.

1.1 Motivation

It is clear that computer networks have since become the backbone of organizations information systems and IT infrastructures. Humans' social activities are becoming fully dependent on IT infrastructures which makes IT the soul mate of mankind currently. Starting from home activities to every business firm and organization , everybody tends to use IT applications directly or indirectly. Consequently this daily interaction of humans with network systems has lead to increased network traffic complexity.[1]. According to Cisco's forecast network traffic will be increasing at an alarming rate where the annual global data center IP traffic will reach 7.7 zettabytes by the end of 2017. By 2017, global data center IP traffic will reach 644 exabytes per month (up from 214 exabytes per month in 2012). Which shows that Global data center IP traffic will nearly triple over the next 5 years. Generally, data center IP traffic will grow at a compound annual growth rate (CAGR) of 25 percent from 2012 to 2017 [2]. Consequently this growing complexity increases the severity of the security threats to our networks and risks the security of the network and computers [3]. This security threat potentially could go to the extent of making high profile attacks other than computer network attacks like military satellites and nuclear power plants [4]. Networks are getting more complex in terms of their scale, topology, and traffic flows. As a result, the inspection and assessment of security incidents becomes a regular business for network and system administrators .

Therefore it is important to apply different security measures like firewalls, antivirus, intrusion detection system and then intrusion prevention system to secure our networks [5]. Network traffic can be detected in two ways either using active detection system or using passive detection system. An active scanning usually generates network traffic and can cause service interruptions as it usually generates network packets and makes use of network band width. Active scanning also requires manual involvement to collect the network traffic data to check open and vulnerable services and for this reason it could be time consuming to actively scan a network system and network traffic. The other method, passive network scanning takes longer time but simply sniffs the network and collects network traffic data passively without sending any packet where the gathered information can be used for further analysis [6] .

Consequently passive network detection and continuous monitoring can give useful information regarding the assets to be protected. This can help network and system administrators address the problem of false positive alarms in intrusion detection and intrusion prevention system. The availability of critical as well as relevant information can limit the number of alerts to deal with, which in turn saves considerable amount of time and resources for security analysts. Due to the continuous change and complexity of networks, it is important to detect changes in a specific network system such as the addition of hosts or services, changes regarding protocol usage or operating system versions. Furthermore new active elements attached to the network can be discovered and registered passively as they join the network traffic [6].

There are many applications that are used to passively listen to the network traffic and gather information on hosts and services they see in the network. To mention some there are pOf, PADS, SinFP and PRADS where PRADS will be discussed more in the next chapters in this paper. These passive scanning tools gather information and put the information in log files. PRADS is selected as the sniffing tool to collect network data passively on this research paper because PRADS has more advantages the other aforementioned passive scanning tools don't have. PRADS is developed by a linux company called *Linpro* and one of the main advantages of PRADS is that it shows network and system administrators about all the services running and the assets residing in their network systems in advance. PRADS also shows the operating systems the hosts in the network are running. Therefore it gives network analysts the the detail about the services and the operating systems on the hosts enabling network mapping. This knowledge can create more awareness in the concerned network analysts and will help them analyse their system and protect any further vulnerability.In addition, this is all done without generating any traffic in the network for everything is done passively and silently [7].

Consequently, network and system administrators have to analyze the actions of the intruder starting from scratch from system logs, where the entire sys-

tem behavior is recorded. Traffic flows are rapid and the number of different applications that are producing the flows get bigger and bigger without limit. Although network traffic is an ideal place to monitor and inspect for security events as all security events have some form of network feature, there is a major problem in that security events can also be hidden among the huge amount of desired traffic. It is often not an easy task to just capture and store network traffic for further analysis. Therefore analyzing and detecting network attacks in almost real-time with the currently available text based tools can be challenging specifically for users with less technical knowledge [8][9].

However, it is believed that humans are excellent at visual processing and finding out abnormal visual trends. By definition visualization is the act or process of interpreting in visual terms or of putting into visual form. But defining it in a more technical way, it is a method or a tool for interpreting image data that is fed into a computer and for generating images or graphs from complex multi-dimensional data sets. Visualization tools can translate the text based network logs into animations that capture the patterns of network traffic in a sufficient way, thus enabling users to quickly identify abnormal patterns that require closer attention and consideration [10]. Such visualization tools enable network and system administrators to filter huge amount of daily network data more effectively than searching text based logs. However visualization of network traffic data is one of the most challenging problems network experts are facing. It has been difficult to graphically represent large amounts of information and be able to show high level of detail for the visual representation to be meaningful [8] [11]. Some of the challenges of visualization could be

- Hard to relate the right data with the right visualization way
- Understanding some of the data syntax and the semantics is difficult
- The size of the log file could be too much

Therefore it is important to use the right technology and approach to solve these challenges. The challenges could be simplified with the help of different commercial and open-source visualization tools.

Network analysis and monitoring is a daily activity by administrator in either network traffic detection or network intrusion detection [12]. This process is always performed to determine what is going on the network. It is very important for administrators to understand and monitor their network very well. Network and system administrators need to manage every service, operating system or the network topology in their system. We deploy network traffic monitoring solutions that scan the network either actively or passively to provide information on the current status of a network [13]. The analysis and monitoring is performed by making log inspections to support the detection of possible breaches and other suspicious network activities.

We visualize information so that we can see the patterns and features that matter to us the most, and depend on that information to make a point to pass a message or tell a story and this allows us to focus on the most important information. It is really hard to simply see and analyze log files as they are, because there are thousands of lines which makes it difficult to analyze using our eye only. So it is a good idea to make use of some visualization tools to analyze the log files and to know what is really happening on the network. Graphical presentation is much more interesting and easier to look into and analyse than textual presentation . That is why it is said that "a picture is worth a thousand words". This project will mainly focus on the visualization of PRADS output data using open-source visualization tools. The idea of this project came from the linux company, *Linpro*, which developed PRADS because the company needs an optimal way of visualizing PRADS log file contents.

1.2 Problem Statement

Considering the current information technology's rapid evolution, growing complexity and distribution of network data, log analysis and correlation has become one of the main tasks of network and system administrators. It is therefore very important to plan and implement a best way to analyze and investigate the ever increasing network traffic data. This master thesis will investigate how to present the network data collected using a passive real-time asset detection system (PRADS) in a way that is easy to understand and analyze. The thesis will mainly focus on the following activities step by step:

- Investigate the contents of PRADS output and make a survey of open source network data and/or log file visualization tools.
- Investigate on how to visualize statically and/or in real-time the output gathered from PRADS using the available open source visualization tools.
- Demonstrate a proof of concept to visualise statically and/or in real-time the host, service and connection data produced by PRADS. This proof of concept will be demonstrated by implementing a prototype solution, by choosing the best free open-source visualization tools to visualise the output which is obtained from PRADS.

1.3 Thesis Structure

The structure of this thesis is as follows:

Chapter 1: Explains the motivation and problem statement for this thesis project.
Chapter 2: Provides background and literature review.
Chapter 3: Gives an explanation about the methodology used, Hardware setup

of PRADS, and scripts used for data collection and automation.
Chapter 4: Provides the results achieved during the research work.
Chapter 5: Explores and analyses the obtained results from the previous chapter.
Chapter 6: Presents the discussion of the whole project and future work.
Chapter 7: Provides the conclusion addressing the main research questions.

Chapter 2

Background and literature

On this chapter there will be a brief presentation with the utilities and terms used throughout the project.Similarly the main visualization technologies and tools for network mapping will also be discussed briefly. In addition, a survey of previous works within visualising host and service data as well as operating system from passive and active network scans will be made and discussed accordingly.

2.1 PRADS

Since the whole project depends on the output manipulated from PRADS, it is worth discussing PRADS and its features briefly. PRADS stands for Passive Real-time Asset Detection System. It is passive finger printing approach used to sniff a network in a silent mode without generating a network traffic. PRADS uses digital fingerprints to recognize services on the network as well as the operating systems, and can be used for network mapping and awareness of possible changes in both real time and static ways. Real-time passive traffic analysis will also make the detection of assets that are just connected to the network for a short period of time possible, since PRADS can gather important information from every packet and saves its output textually in CSV file format that will be able to do anomaly detection [14] [7].

PRADS, unlike other passive asset detection approaches is the most useful approach for passive asset detection, and currently does MAC lookups, TCP and UDP OS fingerprinting as well as client and service application matching and a connection state table. PRADS detects operating systems running in hosts by using different TCP-flag modes like SYN, SYN+ACK, RST and FIN. Various output plugins involved in PRADS can be its default logfile, FIFO, a ring buffer mode, and a logfile which can be created manually. As a result These different possibilities make PRADS a useful replacement for p0f and PADS which are other passive finger printing approaches, hence PRADS will be serving us as the main data source for the network data visualization [15]

[7] [16].

There are many ways to use PRADS. It has many commandline options. Although it is worth refering to the prads man page it has of course different options used for different purposes. To mention some of the common ones are [17]:

Listing 2.1: PRADS Commandline Arguments

```
-i <iface>
        Network device <iface> (default: eth0).

-r <file>
                Read pcap <file>.

-c <file>
                Read config from <file>
 -p <pidfile>
                Name of pidfile - inside chroot

 -l <file>
                Log assets to <file> (default: '/var/log/prads-asset.log')

  -f <FIFO>
                Log assets to <FIFO> -C <dir> Chroot into <dir> before dropping privs.
  -a <nets> Specify home nets (eg: '192.168.0.0/25, 10.0.0.0/255.0.0.0').
  -D Daemonize
```

If you run the prads commandline "*prads -i eth1 -v* ", the assets it sees will be dumped into */var/log/prads.log* and look like this:

Listing 2.2: PRADS Sample Output

```
216.99.152.155,0,6000,6,RST,[0:120:0:*:.:QA:Cisco:LocalDirector (dropped 2)],8,1 394241028
198.50.158.175,0,80,6,SYNACK,[8192:117:1:44:M1460:A:UNKNOWN:UNKNOWN 208.83.20.10 ↙
    ↳ 2:link:ethernet/modem],11,1394240277
94.23.73.143,0,8079,6,SYNACK,[16384:53:1:44:M1460:A:Windows:2000 (2):link:ethern ↙
    ↳ et/modem],11,1394240286
111.253.5.237,0,53,17,SERVER,[unknown:@domain],14,1394241840
177.129.49.101,0,53,17,SERVER,[unknown:@domain],12,1394241927
190.41.54.11,0,17717,17,CLIENT,[domain:DNS SQR No Error],17,1394241932
37.44.162.242,0,53,17,SERVER,[unknown:@domain],9,1394241971
202.133.113.53,0,53,17,SERVER,[unknown:@domain],13,1394242065
178.249.154.30,0,33225,6,SYN,[3036:247:1:40:.:.:unknown:unknown],8,1394242094
82.190.144.36,0,53,17,SERVER,[unknown:@domain],13,1394242138
89.28.87.15,0,53,17,SERVER,[domain:DNS SQR No Error],8,1394242142
115.161.72.154,0,53,17,SERVER,[unknown:@domain],14,1394242186
```

To sort it more this information can be processed further and inserted into MySQL database and so on.

the general format for this data is [16] [15]:

Listing 2.3: PRADS Column Description

```
asset, vlan, port, proto, service,[service-info],distance, discovered

asset = The ip address of the asset.
vlan = The virtual lan tag of the asset.
port = The port number of the detected service.
proto = The protocol number of the matching fingerprint.
service = The "Service" detected, like: TCP-SERVICE, UDP-SERVICE, SYN, SYNACK,MAC,.....
```

```
service-info= The fingerprint that the match was done on, with info.
distance = Distance based on guessed initial TTL (service = SYN/SYNACK)
discovered = The timestamp when the data was collected
```

As mentioned earlier the output obtained from the passive scanning of PRADS is a comma separated format. The scanned data contains information from the fingerprint or signature which PRADS detects passively. The service field indicates the fingerprint type. These can be SYN, SYNACK, RST or FIN depending on the flags set in the TCP datagram, it can be SERVER and CLIENT depending on the type of service detected. Therefore we will be depending on the PRADS output to visualize the IP address, service and OS mapping. Any of the fields mentioned right in the above can be used for our network data visualization [14].

So far we have seen the data collected by PRADS by passively scanning our network and the collected data doesn't show the incoming and outgoing traffic, it simply is collecting the metrics it sees on the network accordingly. On the other hand we can capture network traffic flow data from PRADS as well by running different options like [16] [15].

Listing 2.4: PRADS Network Traffic Options

```
-x  Connection tracking output - New, expired and ended.
-O  (all flow data, per packet)
-L  <dir> :(sguil-style output) or log cxtracker type output to <dir> (will be owned by <uid>)
-B  Log connections to ringbuffer
```

2.2 Database

Database is a collection of logically and coherently related data. A lot of organizations can have tons of data but they cannot have a meaningful data unless they organize their data and make them usable in a logical manner. Databases are everywhere and every one using computer and the internet use databases directly or indirectly but we can never see them. They are hidden behind the tools and services that we use every day. Large companies such as google, amazon or famous social network web pages such as facebook and twitter use databases. Considering our cell phones, our contacts in our mobile phone are stored in a database a lot can be listed on the applications making use of databases.

Databases are not random collections of data , they are rather well organized and structured collections of data. Databases consist of entities, attributes and relationships between the attributes and the entities. The simplest databases are called flat file databases and they store data in column of fields and rows of records.This implies the databases are also a collection of related tables where one can make any operations to manipulate or retrieve data from the databases.In fact the usability of the database is measured by its ease of access

[18].

In database all the knowledge about the data; the way the data is supposed to be or the way the data is related to each other is also stored in the database itself. Therefore the entities in the database and the relationship between the entities are stored in the database where the program or library called the database management system (DBMS)is used to access the database. DBMS is a software or program used to control the database and provides the means of using the database. DBMS only deals within a single table and with the relationship between the entities in the table, it doesnt deal with the relationship between tables.

A decade after the introduction of the DBMS, a Relational Database Management System abbreviated as RDBMS was introduced in order to solve the limitations of DBMS.RDBMS avoids the limitations in old DBMS and introduces Relational model. The relational model has relationship between tables using primary keys, foreign keys and indexes. Thus the fetching and storing of data become faster than the old Navigational model in DBMS. So RDBMS is widely used by the enterprises and companies for storing complex and large amount of data. MySQL is a relational database management system that allows us to put information into a database or to retrieve information from a database. With the highly increasing demand of information by customers and enterprises, database technology is growing and evolving fast, as a result new database technologies like NOSQL are emerging [19].

A database is a separate application that stores a collection of data. Before the introduction of MySQL, each database in the DBMS used to have one or more distinct APIs for creating, accessing, managing, searching and replicating its data. Other kinds of data stores such as files on the file system or large hash tables in memory were also used, but data fetching and writing was not so fast and easy with those types of systems. So nowadays, we use relational database management systems (RDBMS) like MySQL to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys [20].

MySQL is a relational database management system that allows us to put information into a database or to retrieve information from a database. Unlike the oracle database and the microsoft database , MySQL is a free, open-source database engine. MySQL represents an excellent introduction to modern database technology as well as being a reliable mainstream database resource for high-volume applications. Therefore a modern database system like MySQL is an efficient way to organize, and gain access to large amounts of data [21].

As a result we will be applying the use of MySQL database to simplify the visualization of the PRADS output. The use of MySQL database enables us to use the different mysql commands to fetch the columns we are interested in visualizing. Which means the output obtained by running prads on our network will be entirely converted to a database where we will later use it for further visualization. The data to be visualized will be fetched from the database using MySQL commands and used as an input for the visualization tool we are going to implement. More details on how the database is created can be further explained in the approach part.

2.3 Network Topology Visualization

Humans have more tendency to visual presentations than to textual presentations, so the best way to understand complex information is to draw us a picture. The significance of data visualization is very high in order to have a good awareness of our network system. A well represented visualization solves the visibility problem for network and system administrators by effectively expressing the status of the environment we are dealing with, enabling network status analysis in static and real-time data, and making visual link graphs and tree maps any laymen is able to utilize. Good visualization can aid any number of critical measures such as capacity planning, forensics, and root cause analysis [22].

As mentioned in the PRADS section above, the output obtained from executing PRADS resides in a log file in texts with CSV format. Therefore it is really hard to simply see and analyze the log files as they are, because there are thousands of lines which makes it difficult to analyze using our eye only. So it is a good idea to make use of some visualization tools to analyze the log files and to know what is really going on. The look when represented in graphs is much nicer than the CSV textual information. Network topology visualization gives more situational awareness and visualizes status of the system in charge when ever required. Visualization also facilitates communication for it uses graphs and/or maps to communicate with other teams which are interested in the network data analysis. There is not much doubt that graphs are easier to understand than textual events. The customers with little technical background understand visual data better than textual data. That's why a picture tells more than a thousand lines of log files specially for those who are concerned with the most important points and properties of the network [23] . We use visualization to have the following advantages

- Visual representation of textual information.
- Visual display of most important properties
- Reduce analysis and response times
- Quickly visualizes thousands of events

We use visualization using graphs or tree maps when we need to know what is happening in a specific system or business area. That is to make sure what's happening at a specific network, what are certain servers and clients doing , and look at specific aspects of the events. Visualization can also be used when forensics and investigation are needed on a specific system. This can be done by selecting arbitrary set of events for investigation, understanding the big picture and analyzing relationships between the network assets . During visualization log-files or different network events are recorded using a tool for example PRADS in this project's case and parameters like source IP, destination IP, Ports and services are parsed using a parser accordingly and finally visualized in an understandable way using a visualization tool [23].

It is very important to understand the data type of the log file for a visualization designer for the visualization options depend on the type of data on hand. Otherwise the available data to be processed will need to be parsed accordingly in a way it is suitable for the visualization tools.There are many network data visualization tools available both commercial and open-source. In the next subsections we will be discussing those available open-source tools in general. In the next approach chapter, the most suitable visualization tools to visualize PRADS' output will be selected and implemented. In fact a PRADS output visualizer we will be implementing could be anything from a static GraphViz dotfile generator (making static maps) or a java script-driven svg visualizer to a WebGUI or standalone application displaying the changes occurring in real-time.

2.4 Data Formats

The most commonly used data formats used in the network data visualization process by the visualization tools are as follows.
CSV/TSV:
Data format with comma separated values. e.g 12.43.23.45, 22, server, [ssh].
TM3: Data format with the following look.

```
Source port destination Action
12.43.23.45 22 129.89.39.120 failed
```

DOT:
DOT is a plain text graph description language .It is like an assembly language where both humans and computer programs can use it.Dot files have .gv or .dot extentions. It can be represented as follows.

```
digraphs structs {
A->B->C;
B->D;
}
```

GML:
This is the Geography Markup Language which serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet

2.5 Network Data Visualization Tools

Due to the rising network security and complexity challenges, network data visualization has become a very important issue. As a result there has been developed a series of commercial visualization tools and open-source visualization tools. Be it commercial or open-source the main purpose of the visualization tools is to enable the network and system administrators to have good control of their network and simplify their load. The network data visualization tools that will be discussed in this paper are designed and applied in network data visualization system which are partially for users with advanced programming and network knowledge and partially for common users of computers [24].

In addition to the commercial and non-commercial visualization tools, the tools can also be classified as static and real-time visualization tools. The different tools have different functionalities and can give us different outputs for our visual presentations. Some can give us graphs and some can give us tree maps or bar charts. Furthermore, some visualization tools function in a graphical user interface mode and in graphical webs, others work in command line formats or linux shells. The biggest challenge with using visualization tools is using the right sniffing tools to collect the data of our interest and presenting the

data to the visualization tools in a way that enables sysadmins to monitor their system.Therefore the tools that have been surveyed are from all sorts of possibilities and finally the most suitable tools for PRADS output visualization will be chosen.

There are many network data visualization tools which consist of network data and network security visualization tool. To mention some of the network data visualization tools they are Afterglow, Graphviz, Gephi,Cichlid, SeeNet, NodeXL, WatchPoint, Cytoscape, ntop, Nodemap and NAV [24]. Some of these mentioned visualization tools will be discussed in the next subsections. Most of these visualization tools visualize the network data containing IP addresses, ports and services they receive from different network monitoring and sniffing tools. Finally it is worth discussing some of the open-source visualization tools.

2.5.1 AfterGlow

Afterglow is an open source toolkit/application visualization tool developed by Raffael Marty and is used to visualize a text based logfile for the sake of network information visualization. It is a series of perl scripts designed to be used with Graphviz, another visualization tool, to generate link graphs from Comma Separated Values (CSV) formatted files [25]. Afterglow takes CSV files and generates a graph language from the CSV files. Afterglow comprises of two modules1.x and 2.0. Where afterglow 1.x is a toolkit which consists of a collection of scripts written in Perl which can be used to transform the contents of log les into a format which can be used for generating visualizations in the form of link graphs. The link graphs show the communication or relationship between different nodes(machines).The link graph configurations in the afterglow can be reconfigured according to our interest for example the event names, IP address and ports. It can show the security analysts which IP address is talking to which destination address [23].

Afterglow 2.0 on the other hand is a tree map visualization application and shows everything that we see in our network traffic. Afterglow 2.0 is written in Java is designed with only the capability to view log les based on groupings of source IP addresses and intrusion detection log les, but not application log les [26].This version of afterglow can be used to visualize the service-protocol mapping (protocol -¿ service) in our network system. It can be used to analyze the percentage of UDP and TCP protocols and the services they consist of in a certain network traffic [23].

Furthermore afterglow has 3 types of parsers that can parse a log file into a comma separated version file which will be further converted into a graph language. The parsers are written in perl scripts and are pf2csv.pl, tcpdump2csv.pl and sendmail2csv.pl. These parsers parse a given logfile and change it to CSV which will in turn be fed to afterglow. The Afterglow data parsers take a raw input file (tcpdump, Snort, iptables and Argus logfiles), process it and output a comma separated list of records for each of the specific formats accordingly.

The final records are based on the data present in the file and according to the network analyst's interest. Mainly the process is one way of the transformation of the data into meaningful sets for analysis and final graphical representation. Afterglow can work well in conjunction with graphviz, which is another open source visualization tool [23] [25].

When visualizing link graphs we make use of filtering nodes based on the names as well as the number of occurrences on the network traffic data. We can also use different colouring mechanisms for the edges and nodes for the sake of good visualization. As there are thousands of machines connected to our network , afterglow uses clustering to overcome the problem of having many machines. The color and cluster configurations are made in the property file of the afterglow and can be applied to all parameters in the link graphs. Afterglow 1.x has command line parameters represented in the following.

```
-h   : help
-c configfile :configuration file
-d   :print count of nodes
-e   :edge length
-n   :no node labels
-o threshold :omit threshold
-f  threshold :fan out threshold for source node
```

On the other hand after being processed by the conversion scripts or parsers, the extracted CSV files are then fed through scripts that produce one of two formats. Either it generates a DOT attributed graph language file ,the input required by the graphviz library, or it can generate input for the large graphing library (LGL) that is used extensively in bioinformatics. Afterglow preferably accepts either 2 or 3 columns of raw input data to map. For network forensics the 3 column format is the most useful as it can be used to map interaction/events between two entities or attributes within a networked exchange. This operation is particularly useful in tracking or visualizing behaviors that are hard to interpret from text files or represent readily by other graphical or statistical methods [25]..

The figure below shows how the visualization process using link graphs in afterglow 1.x works [23][27].

Similarly the visualization process using afterglow 2.0 can be represented as follows [23][27] :

The second version of afterglow which is the java based one is more interactive and can be zoomed in and out into different parts of the tree map. It also provides different colouring properties for the different hierarchy levels of the tree map to show the mappings between services and protocols or operating systems and has the following command line arguments to run the visualization process.

```
-h   :help
-c file  :property file
-f file  :data file
```

Figure 2.1: Functioning of afterglow 1.x

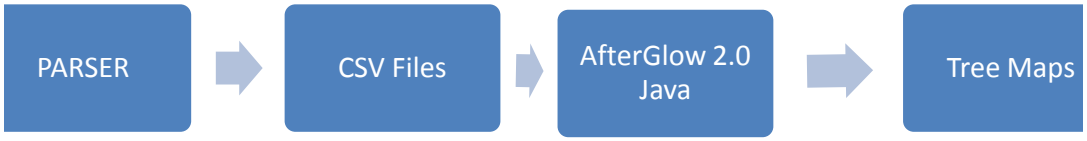

Figure 2.2: Functioning of afterglow 2.0

2.5.2 Graphviz

Graphviz is an open source visualization software/tool and can be run on all operating systems. Graph visualization is a way of representing a log file which is in textual information as diagrams of abstract graphs and networks. It has important applications in different areas such as networking, bioinformatics, software engineering, database and web design. It also has web and interactive graphical interfaces, and auxiliary tools, libraries, and language bindings. The Graphviz layout programs take descriptions of graphs in a simple text language, and make diagrams in useful formats, such as images and SVG for web pages; PDF or Postscript for inclusion in other documents; or display in an interactive graph browser. Graphviz has many useful features for concrete diagrams, such as options for colors, fonts, tabular node layouts, line styles and custom shapes. Unlike afterglow the algorithms of graphviz are meant to deal with static lay-

outs where as for dynamic or real-time visualization there is a modified version of graphviz called dynagraph [28].

Additionally graphviz can work in conjunction with Afterglow and give a nice looking graph. As CSV output is an input for afterglow, it is possible to pipe the fields we are interested in and use pf2csv.pl parser from the afterglow, configure in the property file and feed to the graphviz to make the graphs. For example the outputs could be a combination of the sourceip, destinationip and destinationport fields [23].
Graphviz has four different layout algorithms that are used to make different visualizations. They are dot which generates the hierarchical layout, neato that generates the organic layout, circo which uses a circular approach, and twopi which generates radial layouts. As a result we can choose any of the layouts mentioned according to our point of interest [25].

2.5.3 Gephi

Gephi is another open source visualization tool used to visualize textual data for all kinds of networks and complex systems, in dynamic and hierarchical graphs and enables network analysis. Gephi uses a 3D render engine to display large networks in real-time and to speed up the data analysis and investigation. A flexible and multi-task architecture brings new possibilities to work with complex data sets and produce valuable visual results. Gephi has several features in the context of interactive exploration and interpretation of networks. It provides easy and broad access to network data and allows for spatializing, filtering, navigating, manipulating and clustering [29].
Gephi can be used in the following applications

- Exploratory data analysis
- Link analysis
- Social network analysis
- Biological network analysis

Gephi's developed modules can import, visualize, spatialize, filter, manipulate and export all types of networks. The visualization module in gephi uses a special 3D engine to produce graphs in real-time. This technique uses the computer graphic card and leaves the CPU free for other computing. It can deal with large network (i.e. over 20,000 nodes) and, because it is built on a multi-task model, it takes advantage of multi-core processors. Node design can be personalized, instead of a classical shape it can be a texture, a panel or a photo. Highly configurable layout algorithms can be run in real-time on the graph window. For instance speed, gravity, repulsion, auto stabilize, inertia or size-adjust are real-time settings of the Force Atlas algorithm, a special force-directed algorithm. Several algorithms can be run at the same time, in separate workspaces

without blocking the user interface. The text module can show labels on the visualization window from any data attribute associated to nodes [29].

In order to set up gephi we download Gephi to our computer from [30], install it and open it. The graph file format for gephi can be GEXF,CSV, GDF,GML, GraphML or pajek NET [31]. Gephi is run on a GUI manipulation mode where we specify the number of nodes, number of edges and type of graph. We can also use the mouse to scale the graphs to zoom in and out. Layout algorithms set the graph's shape which is the most important part. The purpose of layout properties is it controls the algorithm and makes the representation of a nice looking graph.There are also ranking modules which enable us to configure node's colour and size. The force atlas algorithm puts into consideration the number of nodes when layouting . Gephi can also be used for network analyses of factors such as betweenness, closeness and clustering coefficient [32]. Gephi can run on windows, Linux, Mac OS X running Java 1.6 ,but the drawback is can only be used for GUI manipulation [29].

2.5.4 Cytoscape

Cytoscape is an an open-source network visualization software platform which is used to visualize different components of a network data. It can also be used for visualization of biotechnological relationships such as gene expressions and different bio molecules. Cytoscape has additional features like plugins. The plugins can be used for network and different molecular analyses. Cytoscape allows us to explore networks relationships and similarity interactively in real-time. It is a very multidimensional visualization tool which can be used for multiple purposes. It is a java based visualization tool [33].

In general Cytoscape is a suitable tool to visualize and analyze any type of data with relationships and interactions among the data using network graphs of any kind involving nodes and edges . This is possible because Cytoscape has a good software architecture which makes use of plugins for specialized features. The plugins can be developed by core developers and the user community. It is still under development and many dynamic features of it are being developed. In fact this tool was originally developed in the Institute of Systems Biology in Seattle in 2002. The latest version of Cytoscape is version 3.0.2 and was released in august 2013 [33] .

In addition cytoscape has attribute values that are used to control the visual aspects of nodes and edges of the graph such as shape, color and size . Also the attribute values can be used to perform different network searches, visualizing network topology ,filtering operations and other network traffic analysis. This tool provides different options to visualize and analyze network data. It can be used by users with ordinary knowledge or for non-programmers using Custom

node graphics and at the same time it can also be used by advanced users for advanced transformations and combination of network data sets using attribute equations [34].

2.5.5 NodeXL

NodeXL (Network Overview for Discovery and Exploration in Excel) is an open-source visualization tool. NodeXL is a useful and straight-forward interactive network traffic visualization and analysis tool that uses MS Excel application as its environment for generating generic graph data, making complex network analysis and visual investigation of networks. NodeXL is supposed to be easy to implement for existing users of Excel, making use of common spreadsheet capabilities such as filtering, sorting and creating formulas. NodeXL extends the spreadsheet into a network analysis and visualization tool by incorporating a library of basic network metrics such as degree, centrality measures, elementary clustering for better graph visualization features [35] [36] .

In the same way we draw graphs and charts in MS Excel, data can be entered or imported into the NodeXL template and it will quickly be displayed as a graph. It is specially meant to help the increasing number of community analysts who have neither the time nor desire to practice through static visualizations or to experience complex programming interfaces. NodeXL is used for static network visualization and network analysis. NodeXL is still an evolving open source project, and it is not meant to be only for normal users but for users with programming skills too. The tool includes an Excel template for simple exploitation of graph data. It is usually used for analyzing social networks like facebook and twitters. The tool looks like the MS Excel as it has a plugin that works with excel and it has also different tabs for customizing the graphs [36].

NodeXL workbooks contain four worksheets: Edges, Vertices, Groups, and Overall Metrics [35] . In the excel sheet we can enter each node one by one or we can import the network data file and we can choose the directed or undirected graphs too. Where the nodes are entered as vertices and we can give them labels, colors as well as sizes for the nodes. We can also select the nodes and move them around to whatever distance we want and use different graph matrix functionalities which enables manual editing of the graph. What makes NodeXL so special is its ability to apply a variety of visual features to the network data elements based on the various individual services and field metrics to be analyzed and visualized. In addition the tools ability to couple spreadsheet and graph visualization, and the more possible ways of its entertaining to allow the different metrics and attributes to be interpreted onto graphs makes it very useful tool among the other network data visualization tools [36].

2.5.6 Cichlid

Cichlid is a distributed, animated network data visualization tool which is written in C programming language and uses the OpenGL and GLUT graphics libraries and it is used to visualize and animate network data sets in bar charts and graphs. The Cichlid code is portable and it is platform independent where it can be applied in Linux, Microsoft windows, FreeBSD, IRIX platforms [37] .Cichlid is a visualization tool for visualizing clients and servers which functions with remote data output and machine autonomy so as to enable the user to view real time network data in 3D visualization. The data being visualized is transmitted from servers to the client engine [37]. The servers will be providing the raw data using TCP connections to the client on which the users will be observing the visual presentations.

Cichlid presents the user with a kind of high-quality 3D, animated visualizations of a wide range of network data presentation [38]. Cichlid is used for real-time, interactive, dynamic visualization of data but not used for static analysis. It just presents the network data to users in a way which seems as if the animations are physical objects. This tool provides network and system administrators with some sort of new insights with the data they collect which makes them analyze their data more. Cichlid presents its visualization in two ways namely 3D bar charts used for showing numeric values and vertex/edge graphs usually used for visualizing network topology [37].

This particular visualization tool follows different functionality procedures to present the network data in a user interactive way. The first functionality is the abstraction and modeling where the real world data is represented in an isolated and independent manner. The second functionality of this tool is data collection and distribution which enables the data to be normalized and passed to the visualization phase. Finally the third functionality is the end product which is the visualization where the data collected will be presented to the users in an interactive way [37].

Summary of Free Visualization tools for Security Analysis and Monitoring

The network data visualization and monitoring tools are summarized in the very next table to give a general view of the purpose of each tool and where they can be obtained from [39] [40] [41] and [42].

Name	Notes	URL
Afterglow	Graph visualization	afterglow.sourceforge.net
Graphviz	Graph visualization	http://www.graphviz.org/
Doomcube	3D IP address and port visualization	www.kismetwireless.net/doomcube
Etherape	Network graph visualization	Etherape.sourceforge.net
INAV	Visualization of network bandwidth, source and destination nodes	Inav.scaparra.com
Google Chart API	Allows creation of dynamically generated charts	code.google.com/apis/chart
Gephi	Graph visualization	https://gephi.org/
Packet Hustler	Network trafc visualization	shoki.sourceforge.net/hustler
Cytoscape	Network data visualization	http://www.cytoscape.org/
RUMINT	Packet-level snifng and visualization	www.rumint.org
NodeXL	Chart visualization of network data	http://nodexl.codeplex.com/
Cichlid	Animated network data visualization	http://newton.tmd.ac.jp/

Table 2.1: Free network visualization and Monitoring Tools

2.6 Previous Work

There have been many network security related papers studying about network monitoring and detection tools as well as log file visualization tools for improved log analysis. The researches on applying information visualization techniques to network security analysis specially log analysis and correlation are increasingly applied to help security analysts manage and monitor their networks. "LogView: Visualizing Event Log Clusters" [26] studies on how visualization can be customized using clusters for improvd log analysis to make the task of network analysts easier. There have been some international presentations and conferences on the visualization techniques to present data interactively for users and security analysts. As a result [13] presents an architecture which enables data-sharing between computer security and network traffic visualization tools. Many of the research papers give emphasis on analyzing the network traffic to crate network awareness for network analysts and users. Research papers like [12] presents about a specific tool for investigating sophisticated network events used for network analysis visualization.

Flamingo: Visualizing Internet Traffic [11] This paper describes a set of visualization techniques that are able to help the task of monitoring and analyzing a network by representing network traffic information in a manageable and interactive way. The software tool that can be used to explore Internet traffic flow data, Flamingo, is able to process live Netflow data in real-time and present a set of interactive visualizations and related processing tools that aid users in network data analysis. Flamingo is made of a server and a client component. The Flamingo server is responsible for receiving raw Netflow feeds from devices in the network that can sample traffic, and then sending processed information to the client for visual presentation. This visualization technique is similarly presented in [37].

"Visualisation of Honeypot Data Using Graphviz and Afterglow" [25] This paper investigates the how data of a honeypot is visualized using afterglow and graphviz and how analysis is made from the visual presentation of the complex textual data. Another paper which studies about "Visualizing Firewall Log Data to Detect Security Incidents" [43] is based on the idea of AfterGlow developer Raffael Marty to visualize the firewall log data and implements afterglow as the visualization tool. Similarly in [**?**] it is described how network data visualization helps to pick malicious patterns easily.Similarly in the research paper in [26] it is described how clustering simplifies the visualization and analysis of large logfiles which can be complicated and difficult to understand in case the logfile size increases. It explains and shows that clustering is the best solution to handle the increasing size of log files.

There are many papers discussing the possible ways of visualizing network data, but there are few papers explaining PRADS. Petter Bjerke explains how passive detection of OS is done using PRADS [7]. He further explains the difference between pOf and PRADS OS detection. Similarly Mats Erik discusses "Passive Asset Detection using NetFlow " [44]. In this paper Mats Erik explains how passive asset detection system using NetFlow data is highly scalable and capable of processing a lot of data. Finally Jostein Haukeli:" False positive reduction through IDS network awareness" [14] discusses how PRADS creates network awareness for system and network analysts by reducing false positives.

Chapter 3

Methodology

"Research is to see what everybody else has seen, and to think what nobody else has thought." -Albert Szent-Gyorgyi

Chapter 1 and chapter 2 presented a detailed explanation of PRADS and the network data visualization tools. Using PRADS, network and system administrators can get good awareness of their network system for PRADS reveals services which the network and system admin was not aware of in advance, along with the operating systems hosts are running. This special functionality of PRADS is the reason why PRADS is used as our data collection tool. Using different network data visualization tools the output found from running PRADS on the required network system can be presented to different users or network security analysts in the form of graphs, tree maps and different animations. This visual presentation of network data obtained from running PRADS will make the log analysis easier and better.

In order to implement the visualization of PRADS output there are a number of open-source network data visualization tools which are explained in chapter 2. According to the problem statement a survey and research has been made on similar projects to this particular thesis. As a result an experiment environment is established for running the PRADS and enabling data collection. In addition PRADS software is installed and configured fully on the experiment environment and PRADS and its output syntax and semantics will be investigated in detail.

This research work is all about presentation of data particularly network data to users in the best possible way which is a simplified form of data presentation and on the way making the task of network and system administrators easier. Therefore this paper demonstrates on how we present a certain knowledge/analysis to a normal user or security expert where it will be easier to understand and deduce the pattern by looking at the visual presentation than looking at

each line of a textual presentation of data. As a result this research paper is an exploratory research that investigates on how to visualize PRADS output data using open-source visualization tools. This paper tries to look into how some of the mentioned network data visualization tools function and in what platforms they work better. Therefore the visualization tools are compared with one another on their suitability and ability to best visualize PRADS data accordingly which makes this research paper a mix of exploratory and comparative studies.

This chapter presents the experiment environment setup design, system requirements, data collection strategy, database configuration and the proposed visualization methodology setup to visualize the network data. This is the approach to visualize the network data output obtained by running PRADS on a certain network. In fact it seems that the project needs somehow advanced technical ability to visualize the PRADS data as it is intended to show the network and system administrators. Therefore due to the newness of the PRADS detection tool and the semantics and syntax of its output which creates some complexity difficulty, it is expected to encounter some technical difficulty while doing this research paper.

3.1 System Requirements

PRADS will be run in a gateway with the following requirements in a virtual lab environment.

Item	Description
FQDN	masterfw.vlab.cs.hioa.no
Machine model	Quad-Core AMD Opteron(tm) Processor 2376
Processor	CPU 2294.332 MHz (Dual core)
NIC	2
Operating System	Ubuntu 12.04.1 LTS x86 64
Linux kernel	3.2.0-36-generic
RAM	630884 kB

Table 3.1: System Requirements

3.2 Experiment Environment Setup

The goal of this thesis is to find the best way to visualize the output network data obtained from running PRADS in a network and listening to a network traffic so that network analysts will get better help from the visual presentation. It is believed that our eyes are excellent at recognizing visual patterns

than many lines of textual data. Therefore the experiment environment is set up in a way that enables PRADS listen to a network traffic and collect what it sees in the network passively and logs its content to a log file where the log file will be further used for the visualization process to finally give visual presentations in graphs, tree maps or bar charts.

For network and system administrators it is really important to know our network in advance. It is necessary to know the subnets of our network so that we will be able to recognize which one is the external network and which one is the internal network from the PRADS output and this in turn will help us have a good visualization plan to make for future network mapping and analysis. In addition when we have good control of our network we can manage to make some automated processes to check if services can be recognized by PRADS. For example we can start and stop some services in our network and see if PRADS does detect this.

The experiment environment in the next figure is a virtual lab environment in the university college of HIOA consisting of a gateway *gateway1* with external interface Eth0 and internal interface Eth1. PRADS is installed and configured in the gateway1 and data can be collected passively by running PRADS in either of the interfaces according to our point of interest. PRADS usually detects TCP packet services but it can also detect UDP packets like DNS. It detects what it sees in the network traffic. Therefore it is necessary to generate some network traffic activities from the internal machines deliberately to create some internal activities. This can be done by writing some scripts that create network traffic or scripts that start and stop some services. This process is important when it is required to run PRADS in the Eth1 because there may not be a lot of traffic in this specific network topology. But for the case of running PRADS in the Eth0 (the external interface) there is a lot of traffic movement and there is no need of making deliberate traffic generation. This data collection process will be extended to the whole network of HIOA and the Alto-Openstack.
The following setup is our virtual network experiment setup.

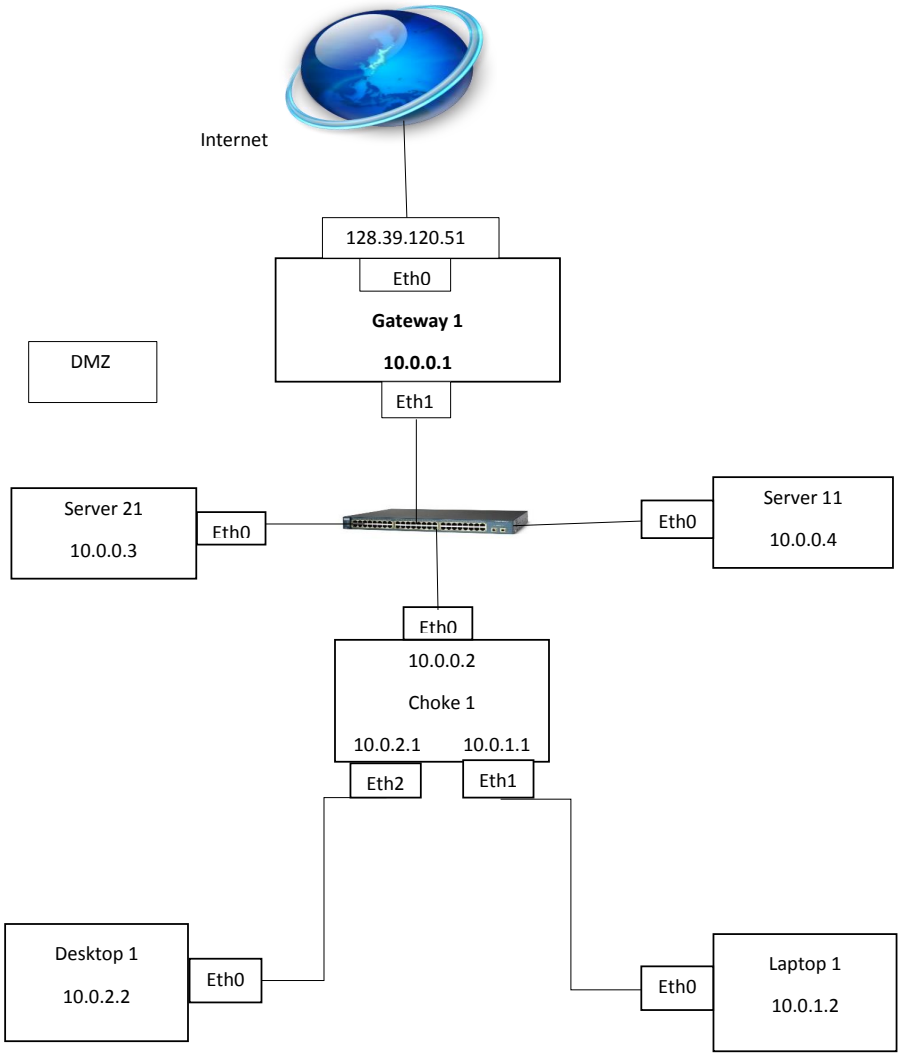

Figure 3.1: Experiment environment Setup

In the above network set up we have three different internal networks with 10.0.0.0/24, 10.0.1.0/24 and 10.0.2.0/24 networks.

Although this virtual lab environment is not a big one, this is just for the sake of testing how PRADS output can best be visualized using the available visualization tools. PRADS has been tested in previous works in larger networks. The visualization process can also be tested in larger networks than this network set up according to our point of interest. Therefore it is part of the plan of this project to test the visualization process by considering the network data collected from HiOA and the Alto-Openstack using PRADS.

3.3 Visualization Tool Selection

A survey has been made on how the different visualization tools function, the platforms they use and their ease of use according to the first part of the problem statement. Most of the visualization tools investigated are open-source visualization tools where the tools are free for every body to use. As a result some of the open-source visualization tools will be investigated in relation to their usability to visualize PRADS output data. In this paper, as far as PRADS output is concerned the ideal tool to visualize its output is Afterglow for its simplicity and ease of use for automation. Infact Afterglow has two versions the perl version (Afterglow 1.X) and the java based version (Afterglow 2.0) which is used to visualize hierarchical graphs of different depth but for the sake of simplicity in the prototype development we will be using the perl based Afterglow [45].

The output obtained from PRADS is in the form of CSV files and the Afterglow has parsers that pars a certain log file into CSV files because Afterglow takes CSV files as its input to visualize them. That's why Afterglow has become the ideal tool to visualize the data collected using PRADS. Afterglow will be implemented in the proof of concept demonstration in conjunction with Graphviz, another open-source visualization tool which has different graph layouts that enable us look into the network data from different angles and different visions. Therefore due to their simplicity and their automation capability Afterglow and Graphviz will be implemented in the prototype development. Both tools have good customization options that increase the usability and understandability of the visual presentation for users and network analysts. Graphviz has different layouts that make the visualization process flexible and applicable. Similarly Afterglow has properties file options to make some clustering and colouring methods. The clustering and colouring methods make the visualization process easier and more purposeful [27].

3.4 Proposed Visualization Methodology Setup

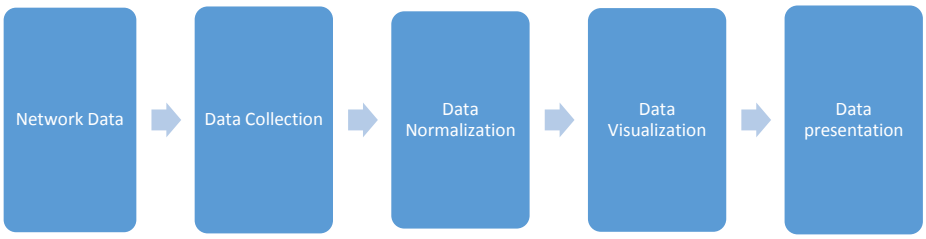

Figure 3.2: Visualization Methodology Setup

3.4.1 Data Collection

For the sake of visualization data can be collected either in active scanning way or passive scanning ways. With active traffic scanning we have active scanning tools like nmap or Nessus and the scanning is done through sending multiple probe requests and recording the probe responses. But with passive scanning tools like PRADS Records and identifies traffic seen on a network without actively scanning a system. Passive scanning tools do not send traffic to detect network assets. The tools just record what they see on the network to a logfile.

PRADS (Passive Real-time Asset Detection System) is the program that is used to collect the data by passively listening to the network traffic, and uses the information gathered to map the network [16]. PRADS passively reports everything that it sees on the network including all the hosts that are in the network including the services the hosts are running. PRADS does use the protocols ICMP, TCP and UDP traffic to detect the network data including the IP address of each asset, ports being used, etc. PRADS is a very powerful tool and it has the ability to detect hosts based on ARP packets.

The output from PRADS can be put in a logfile where the default logfile is */var/log/prads-asset.log* or in a **FIFO** device to send the results to another application or can be put into a database for further use. In addition to logging to logfiles to collect the information from PRADS, There are developing new ways for the upcoming PRADS releases, we can have access to assets via shared memory using the ring buffer options in the prads options. That will make it easier for extracting information from the running PRADS process that is currently running. PRADS also has a perl script prads2db.pl which parses a prads asset log-file and inserts the info to a DB so we can query it for information using different MySQL commands. Generally the output is collected in a CSV

format. The format has usually eight (8) fields namely asset, vlan, port, protocol, service, service-details, distance and timeslot(discovery time) orderly. The asset field usually refers to source IP but it can also sometimes be a destination IP too.

The data collection process depends on which data we are interested to visualize, for example whether we want to figure out the outgoing traffic from our internal network or the incoming traffic to our internal network. This will enable us to focus on our point of interest and reduce the amount of data we must look into. Similarly When collecting data In order to identify which subnet we are dealing with and which asset belongs to our network and which one is external and internal we need to run PRADS by specifying our home network it should look into. For example we ran PRADS as follows:
prads -a 128.39.120.0/24, 10.0.0.0/255.255.255.0
In addition we can check if PRADS really scans every service running by executing some services on the specified network.In this case whenever we run services from outside to the specific network like ssh services,the details will be displayed in the PRADS logfile.

3.4.2 Data Normalization

After data is collected, it can not be simply passed to a visualization tool and get presented visually. It needs to be parsed according to our interest. We need to rectify the fields we want to visualize. The data needs to be customized in a way that is understandable for the users and analysts of the data as well as for the visualization tool. As a result we can use some scripts to pipe or take out the required fields that we want to visualize and put the parsed file as an input for the visualization process. Actually the parsed field are also CSV files.

Therefore, we need to start normalizing the data, we need to extract the data that we have like the end nodes source IP, source port, destination IP, destination port, the services or other fields of interest. We can use some perl scripts, or some commands like pipe, grep, awk to pull out those fields. It is also possible to use some ready made parsers that enable the same job like the perl script or the mentioned linux commands. While parsing it is a good idea to remember or give focus on who is the client and who is the server, which is the source and which is the destination. It is important to note who is initiating the communication by looking at the SYN, SYNACK packets as the clients usually send the SYN packet and the servers also reply with SYNACK as a means of three way handshaking and connection will be established consequently.

In fact there are some PRADS options that help us keep track of the connection tracking. Even if the final result is logged into the log files in the same format of PRADS default options "prads", there are options that show the connection

tracking on the screen. The options are as follows.
prads -i eth0 -O... gives the following output

```
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
128.39.120.51:22 -> 37.191.132.112:54916
```

prads -i eth0 -x...gives the following output.

Listing 3.1: prads -x

```
1398211438000000243|2014-04-23 00:03:58|2014-04-23 /
00:03:58|0|0|128.39.120.29|0|224.0.0.251|0|1|8|0|0|0|0|[expired]
1398211399000000242|2014-04-23 00:03:19|2014-04-23 /
00:03:19|0|6|128.39.120.8|30296|198.100.150.99|53|0|0|1|24|0|18|[expired]
1398211778000000251|2014-04-23 00:09:38|2014-04-23/
 00:09:38|0|6|198.100.150.99|53|128.39.120.156|30444|1|24|0|0|18|0|[New]
1398211779000000252|2014-04-23 00:09:39|2014-04-23/
 00:09:39|0|6|198.100.150.99|53|128.39.120.133|30421|1|24|0|0|18|0|[New]
1398211804000000253|2014-04-23 00:10:04|2014-04-23/
 00:10:04|0|6|93.174.93.51|48027|128.39.120.136|4438|1|20|0|0|2|0|[New]
1398211811000000254|2014-04-23 00:10:11|2014-04-23/
 00:10:11|0|6|192.95.20.134|53|128.39.120.126|30414|1|24|0|0|18|0|[New]
1398211814000000255|2014-04-23 00:10:14|2014-04-23/
 00:10:14|0|6|198.100.156.138|53|128.39.120.176|30464|1|24|0|0|18|0|[New]
1398208938000000003|2014-04-22 23:22:18|2014-04-23/
 00:09:13|2815|17|0.0.0.0|68|255.255.255.255|67|92|27076|0|0|0|0|[expired]

1398211749000000249|2014-04-23 00:09:09|2014-04-23/
 00:09:12|3|6|82.221.102.179|58323|128.39.120.51|22|14|1655|14|2447|27|27|[ended]
```

Showing the *new connections, expired ones, and ended connections.*
So we can have outgoing traffic as well as incoming traffic accordingly. In our case when we use eth0 as our interface for prads it shows the the source ip addresses and their ports that are trying to access the internal network.

Generally with normalization we can normalize the data into different formats of files according to the visualization tools we will be implementing. Some of the data formats are mentioned in the background chapter section *Data Formats.* But in the case of this research paper we will be applying Afterglow as our visualization tool and Afterglow takes CSV files as input and visualizes the data in different forms of graphs. Afterglow has 3 types of parsers that can parse a log file into a comma separated version file which will be further converted into a graph language. The parsers are written in perl scripts and are pf2csv.pl, tcpdump2csv.pl and sendmail2csv.pl. These parsers parse a given logfile and

change it to CSV which will in turn be fed to Afterglow. The Afterglow data parsers take a raw input file (tcpdump, Snort, iptables and Argus logfiles), process it and output a comma separated list of records for each of the specific formats accordingly. Fortunately the output from PRADS is already a CSV format and all we need is to pick out the fields we are interested in visualizing, there is no need to to change the output to another CSV file.

3.4.3 Data Visualization

After data is collected using the passively sniffing tool, PRADS, it will be normalized according to our point of interest. The normalization can be done by parsing the required fields using a script, or some other linux commands like pipe and grep. But in the case of the network data visualization we are going to apply on this specific project, where the visualization tool to be implemented is Afterglow, We will use a parser in the Afterglow in the properties field. The properties field in the Afterglow is important for selecting which fields we will use, what colour we will use for each field and how the clustering is done. The properties field is executed together with the script we will implement for our prototype.

Therefore the network data visualization will go through first data collection, then the normalization or data parsing, and finally the visualization process comes. A script in perl will be developed integrating all these procedures at a time where the script will run PRADS on a certain network and then the script will pick specific fields according to the user's interest and pass it to the visualization tool. The same script will make the visualization tool draw a graph and present the graph to the users or network analyst. The visualization procedure can be:

- Visualize the service and subnets in our network
- visualize what is the incoming network/traffic
- visualize what is the outgoing network/traffic

3.4.4 Data Presentation

Data presentation is the final result of the network data visualization process. The data to be presented needs to be as simplified as possible. It is from the presented data that conclusions are drawn. Special patterns or irregularities are easily identified from visual presentations as human eyes are excellent at recognizing unusual patterns in the network. Data can be presented in graphs or tree maps in case of Afterglow. Afterglow has different layouts that help us customize our presentations. But the most important specifications for the

presentation are made in the properties file. We can use different colouring mechanisms for the edges and nodes for the sake of beautiful visualization. As there are thousands of machines connected to our network, Afterglow uses clustering to overcome the problem of having many machines in a congested visual presentation. The color and cluster configurations are made in the property file of the Afterglow. It is in this properties file the size of each edge and each node is specified accordingly. It is also possible to make specifications for showing IPs of each subnet with the services running with fading colors based on time of stay in real-time. With the older ones more faded and clustered IPs of the same subnet in one node.

3.5 Database Configuration

There are many options on how to make use of the data collected by running PRADS on a certain network. The output data can be logged to a default logfile or to a manual logfile. It can also be used in a shared memory using the ring buffer options in the prads options. PRADS also has a perl script *prads2db.pl* which parses a prads asset log-file and inserts the info to a DB so we can query it for information using different MySQL commands. As a result it is necessary to install and configure a *MySQL-server* in the system PRADS is running. In our case we created a database named as *'prads'* and created as well a user with username *'prads'* and password *'prads'* for the database that would be created from running PRADS in our network. As a result we can just use the same database and use the MySQL commands to select the columns we want to visualize from the database and then we will visualize our network data.
Procedure to create PRADS output database [7] :

Listing 3.2: Creating PRADS Database

```
1 tcpdump −w tcpdump.pcap.....creating the pcap file
2 prads −r tcpdump.pcap −l bench.....making prads to read the pcap file into the logfile name bench.
3 ./prads2db.pl −−file /home/group1/bench...to write the records into a database
```

After the installation and configuration of the MySQL-server, the database created using the **prads2db.pl** can be accessed as follows:

Listing 3.3: Accessing PRADS Database

```
1 mysql −u prads −h localhost −p
2 USE database−name; where database−name is prads in our case.
3 SELECT ∗ FROM prads; to select all the querries of the database
4 SELECT ip,port,service FROM prads; to select specific columns of the database
```

These specific fields could further be visualized using visualization tools if it is necessary.

3.6 Alternative Approaches

The approach that will be applied in this master thesis is the methodology described so far. But there are other approaches that can be implemented to solve the same problem statement. The problem statement suggests to use real-time visualization too if it is applicable in the given period of time. One of the alternative approaches could be by creating a database where PRADS will insert its output into. The next step is using MySQL query commands to pick out the required columns for our visualization tool to visualize the data in graphs. Which implies the output from PRADS goes to the database where the database in turn will be the source of input for the visualization tool. Therefore it is a good idea to make use of databases and it can be implemented in the following way.

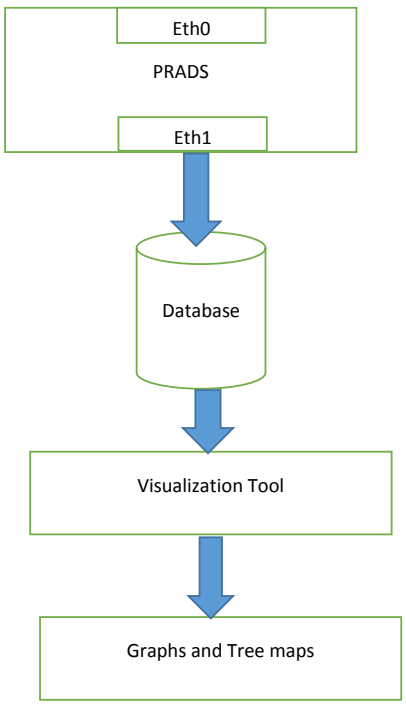

Figure 3.3: Database Based Approach

The second alternative approach is little bit advanced and is used mainly for real-time visualization of the PRADS output.This release of PRADS supports a ringbuffer log, which can be used to write log directly to a visualising agent through shared memory where both PRADS and the visualization tool use the shared memory. This is possible because PRADS supports the -B option in order to log connections to ring buffer, in addition to the other option -f fifo argument and the 'fifo: /path/to/fifo' configuration option to feed events into a FIFO [16]. The process for the second alternative approach can be shown pictorially in the following figure.

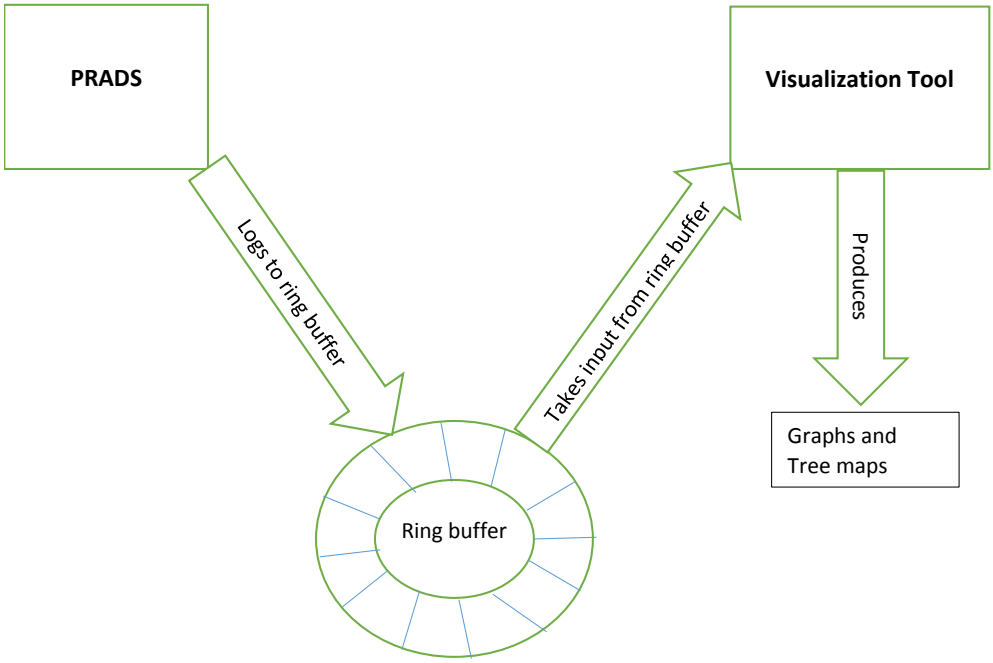

Figure 3.4: Shared Memory Based Approach

3.7 Implementation

Finally after deciding on which approach to work with and after making enough survey on the available open-source visualization tools; each tool will be investigated how it can be used to visualize PRADS output. After full consideration and investigation of the available open-source visualization tools a tool which is suitable to PRADS output visualization will be selected. As a result a prototype will be developed using a perl script using the selected visualization tool to demonstrate a proof of concept to visualise statically and/or in real-time the

host, service and connection data mappings produced by PRADS.

3.7.1 Prototype

The prototype will be developed using a perl scripting language and it is planned to automate everything including the data collection, parsing, normalization and visualization. The prototype will incorporate the data collection tool which is PRADS in this research paper's case and the visualization tools *Afterglow* and *Graphviz*. To check the efficiency of the data collection tool during our prototype development, we can use some scripts or command lines that generate/run and stop services automatically or manually.This is supposed to test whether the services running in a host or newly installed services on hosts are detected by PRADS or not. The prototype consists of two perl scripts namely *parser.pl* and *visualizer.pl*. The parser script filters out the required fields out of the eight column PRADS log file and pipes the fields to the visualizer script which makes use of the visualization tool Afterglow. The visualizer script, using Afterglow, produces graphical language which can be read by the different Graphviz layouts and finally a graphical presentation *(.gif)* is produced. The prototype is designed to work according to the following diagram [46].

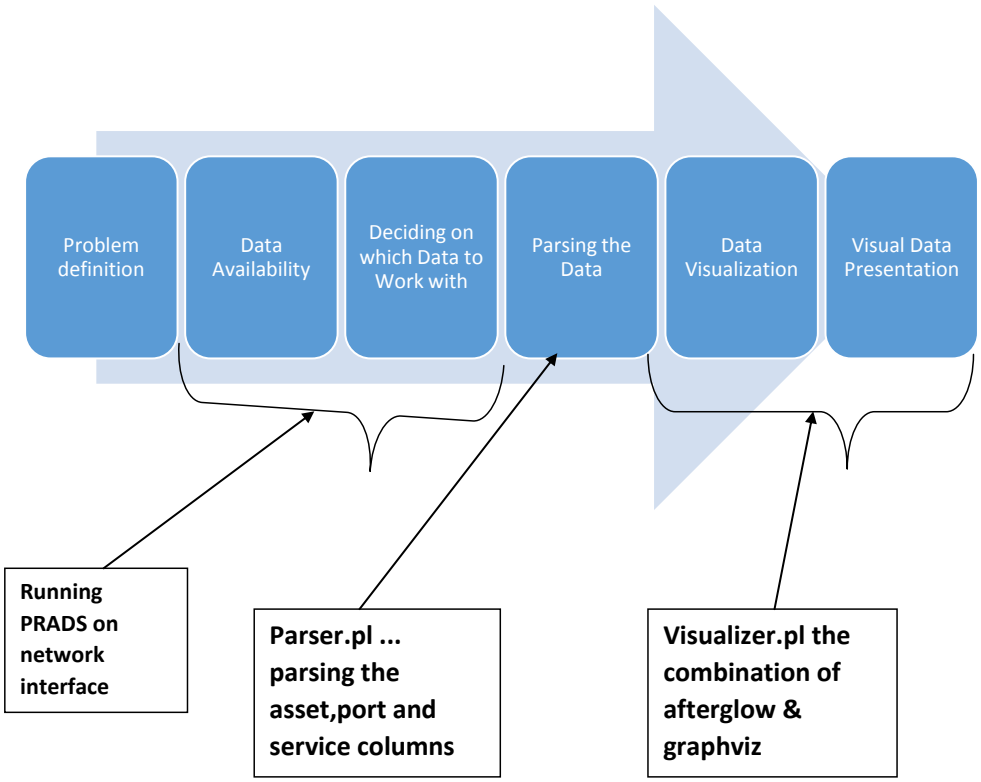

Figure 3.5: Prototype Implementation

3.7.1.1 Problem Definition

Before doing anything it is important to first note the problems which will be solved by the prototype development. The main bottlenecks on the network system should be identified. It should be first planned what questions and problems the will the program solve. Knowing the real problem solves time and energy that will be wasted during the whole process of the prototype implementation. In the case of this research paper the problem can be there are services running on a certain network which can be a source of vulnerability to external intrusions where the network and system administrators are not aware that the services are running. Which leads to the need to tackle such a problem.

3.7.1.2 Data Availability

After knowing the real problem it is important to know how to get the real data to deal with. It is necessary to identify what type of network data to deal with and what the domain of the data is. If data is already available then go

to the next phase or else devise a means to collect the required data. There are many ways of collecting data. It could be active data collection or passive data collection. But in the case of this prototype development the data is collected using a passive data collection tool, PRADS. In this prototype development the data is collected by running PRADS in a controlled way. Traffic is intentionally generated by making different activities in the computers of the local network like starting and stopping services, using ping and ssh commands, and using scripts that generate traffic continuously. As a result everything will be detected and reported by PRADS.

3.7.1.3 Data Filtering

The data available to be executed can be too much to deal with. It is important to limit the amount of data that will be dealt with. The whole data may not be required for further processing and it is a good idea to begin with smaller data first and upgrading the amount of data bit by bit. In the case of PRADS, it collects every thing it sees in the network which results a lot of file to work with in the log file. Therefore the data collection should be limited to a specific network by specifying PRADS to collect data on a specific subnet, or on the interface where the problem to be solved exists. That is in case the internal network is the point of interest PRADS should be run in the internal interface Eth1 of the gateway gateway1 which is shown in the Experiment Environment Setup section figure. By default when PRADS is run in a network it sniffs on the interface Eth0. Therefore a decision should be made before hand.

3.7.1.4 Data Parsing

It is in this section where the data normalization starts. The data format could be any format with many fields and columns. But the data needs to be normalized in accordance with the visualization tool in charge. The visualization tools that will be implemented in this prototype testing is Afterglow in conjunction with Graphviz. The Afterglow visualization tool will be creating a graphical language first that will be converted into graphs and tree maps by the Graphviz visualization tool. Therefore a perl script will be developed to parse the data that will be considered for the visualization phase and fed to the Afterglow visualization tool.

3.7.1.5 Visualizing the Data

As long as the data is parsed and ready, it will be piped to the visualization tool, Afterglow, for further processing. Afterglow will give the graphical language which will be read by the Graphviz visualization tool. It is in this phase that the real customizations are made. Properties such as clustering and colouring

and other filtering mechanisms are made. Afterglow has properties file which enables different customization possibilities according to the user [23].

3.7.1.6 Presenting the Data

Data presentation will be a visual presentation in the form of graphs and tree maps. The visual presentation is supposed to help rapid network data and traffic analysis. The network mapping between assets and services and each connection data will be displayed. Graphviz has different graph layouts namely *dot, neato, circo, twopi* which give many options for visual presentations.

3.7.2 Properties File

The properties file is the script shell where the real customization of the graphs to be displayed is made. It is in this configuration file that the filtering of the nodes to be displayed and those to be left out in the graph is made. Before visualization is made a decision needs to be taken on where emphasis should be given. We have to plan and identify how we need the visualization to be, what nodes should be included and what type of services do we need to visualize in our network. If the filtering and customization is not made, then every textual information in the log file will be visualized in a complicated graph which will make it difficult to understand and make any analysis out of it. To avoid such complexity, the properties file enables us to make even more complicated customizations and filtering. The customizations could be implemented using different colouring methods to the nodes and edges and clustering similar nodes together. Assigning different sizes to edges and nodes, and assigning shapes to nodes is also part of the customization process made in the properties file. Afterglow has a sample properties file but needs to be customized to our needs.

The sample script of a properties file looks as follows and it is by modifying this properties file that every customization is made. Customizations like clustering, colouring, edge size and node size assignments and others are performed by modifying this properties file.

Listing 3.4: Afterglow 1.x Sample Properties

```
1  color.source="yellow" if ($fields[0]=~/^128\.39\..*/);
2  color.source="greenyellow" if ($fields[0]=~/^10\..*/);
3  color.source="lightyellow4" if ($fields[0]=~/^172\.16\..*/);
4  color.source="red"
5  color.event="yellow" if ($fields[1]=~/^128\.39\..*/)
6  color.event="greenyellow" if ($fields[1]=~/^10\..*/)
7  color.event="lightyellow4" if ($fields[1]=~/^172\.16\..*/)
8  color.event="green"
9  color.target="blue" if ($fields[2]<1024)
10 color.target="lightblue"
11 color.sourcetarget="pink"
```

```
color.edge="blue" if (1)
size.edge=1;
#Changing node labels:
#label=substr(field(),0,10)
#URL for nodes (used for Graphviz to enable image map functionality)
#This is an exampel of how to use AfterGlow with Splunk
url=http://localhost:8000/en-US/app/search/flashtimeline?q=%20\N%20starthoursago%3A%3A24
#Using node sizes:
#size.source=1;
#size.target=200
#maxNodeSize=0.2
```

Chapter 4

Results

In this chapter, the results obtained according to the problem statement and the previous chapter,methodology,will be displayed and explained in detail. A survey of open-source network data visualization tools was conducted and some visualization tools were investigated how network data can be visualized and presented to users or network analysts accordingly.Further more the syntax and semantics of PRADS was investigated in detail and many of the visualization tools were investigated and tested with regards to PRADS output. As a result possible visualization and data presentation ways are suggested here. Finally a proof of concept is demonstrated by implementing a prototype solution using one of the open-source visualization tools that is afterglow by automating the prototype using a perl script.

Part I

4.1 Summary of Open-source Visualization tools for Security Analysis and Monitoring

Many network data visualization tools have been mentioned and discussed in the background chapter.In order to address the first part of the problem statement, the tools selected here are with regards to their usability and suitability to visualize PRADS output data.It is worth describing some of the selected and popular visualization tools which can be used to visualize and present PRADS output in a better way.There for 3 open-source visualization tools are selected and briefly discussed in this section as a result of the survey made on the free available tools and their suitability to visualize the logfile from PRADS.

4.1.1 Afterglow

Afterglow is an open-source network data visualization tool developed by Raffael Marty. It has two versions with the first version afterglow 1.x and is perl based, the second version is afterglow 2.0 and is java based.In both cases it takes CSV files and as an input and generates graph languages and then link graphs. Afterglow is used for both static and real-time visualizations.It can be found in [47] and is freely accessed and functions as in the following figure. Afterglow has a properties file for making different customizations for the graphs [48] . The customizations can be clustering of nodes, giving different colors, different shapes and different sizes to nodes. The following figure shows how afterglow works in principle to present data in visual presentation [49].

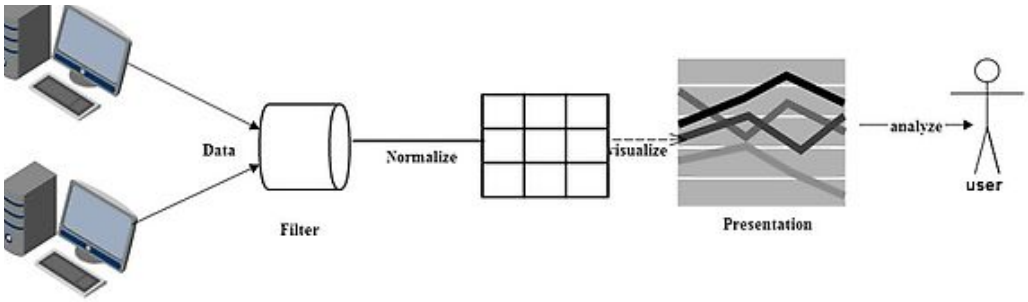

Figure 4.1: Afterglow Functioning

Afterglow has many options to customize and filter the nodes to be shown in the graph. We can filter out nodes based on the number of occurrences shown in the network traffic, and the number of connections a node should have to appear in the graph. The Afterglow options for making different customizations are displayed in the methodology chapter.

4.1.2 Graphviz

Graphviz is an open source visualization tool and can be run on all operating systems. It is used to visualize every kind of structured information in diagrams of networks,graphs and tree maps . Graphviz has many useful features for concrete diagrams, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes. It is mainly used for static visualizations and can work in conjunction with afterglow. Graphviz has different graph layouts which give different visual presentations. The famous graphviz graphical layouts are dot,neato,circo and twopi. Graphviz is freely found for downloads in [50].

The working principle of graphviz can be represented in the following figure 4.2 [51]. The **.dot** file is a graph description obtained from the afterglow visualiza-

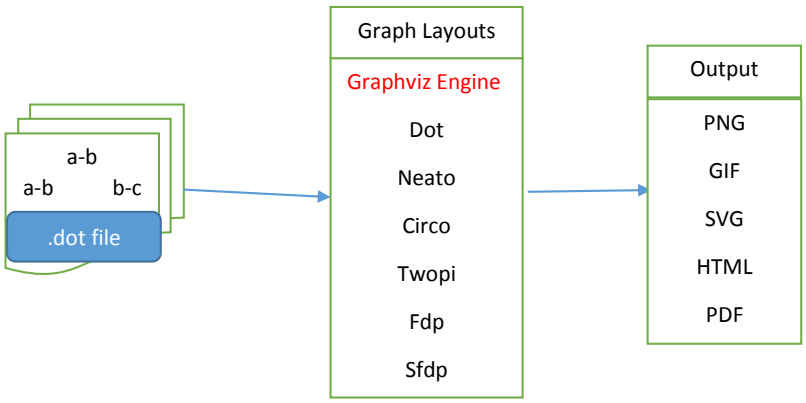

Figure 4.2: Graphviz Functioning Principle

tion tool. This is the input considered by graphviz for the visualization process using one of the layouts described in the functioning principle to give the visual output. The output format obtained using our prototype is the (.gif) format.

4.1.3 Gephi

Gephi is an open source visualization tool used to visualize textual data for all kinds of networks and complex systems, in dynamic and hierarchical graphs and enables network analysis. Gephi uses a 3-dimensional render engine to display large networks in real-time and to speed up the data analysis and investigation. In this visualization tool many customizations are possible and has the capacity of showing as many nodes as possible in a single process. Gephi can run on windows, Linux, Mac OS X running Java 1.6 ,but the drawback is it can only be used for GUI manipulation. This visualization tool can be accessed freely from [52] and usually used for drawing social networks. From this visualization tool we can have a report on the graph like the number of nodes, the number of edges and type of graph. We can make as many customization as possible such as colouring, zooming in and out with a mouse and using different layouts for it is a GUI mode.

Part II

4.2 Possible Methods of Visualizing PRADS Data

PRADS logs its output in eight-column field of "asset, vlan, port, protocol, service, [service-info] , distance, discovered". The main advantages system and network administrators can get from running PRADS in their network is they will be aware of the services running in their systems in advance as well as the operating systems being run in each host observed in the network. PRADS can also give a detailed information on the services running and the hosts running the services such as the uptime, and type of service in the service-info field. The other knowledges the system and network administrators will be provided with is the asset IPs and the ports associated with the assets. In fact it should be noted that the ports are also associated with the services running. It should also be clear that the service-column field has different UDP, TCP-flags, client and server services. But the most important services for network and system administrators will be the client and server services in general. Therefore this section shows the possible ways of visualizing the important fields of PRADS output data to network traffic analysts aimed at helping them know their network topology , and host-service, asset-port, host-operating system mappings.

When PRADS is run in the network for some time, it collects what it sees in the network and when it is ended it reports a lot of information in the terminal. Therefore some of the possible visualization ways of PRADS information depend on this terminal report. The terminal report obtained from running PRADS in eth0 looks as follows.

Listing 4.1: Terminal Output of PRADS

```
[*] prads 0.3.3-1-gf93c93e
      Using libpcap version 1.1.1
      Using PCRE version 8.12 2011-01-15
logging to file '/var/log/prads-asset.log'
[*] Loading fingerprints:
   CS_MAC /usr/local/etc/prads/mac.sig
   CO_SYN /usr/local/etc/prads/tcp-syn.fp
   CO_SYNACK /usr/local/etc/prads/tcp-synack.fp
   CO_FIN /usr/local/etc/prads/tcp-fin.fp
   CO_RST /usr/local/etc/prads/tcp-rst.fp
   CS_TCP_SERVER /usr/local/etc/prads/tcp-service.sig
   CS_UDP_SERVICES /usr/local/etc/prads/udp-service.sig
   CS_TCP_CLIENT /usr/local/etc/prads/tcp-clients.sig
[*] OS checks enabled: SYN SYNACK RST FIN
[*] Service checks enabled: TCP-SERVER TCP-CLIENT UDP-SERVICES MAC
[*] Device: eth0
[*] Dropping privileges to 1:1...
[*] Sniffing...
^C-- prads:
-- Total packets received from libpcap : 1327
-- Total Ethernet packets received : 1327
-- Total VLAN packets received : 0
-- Total ARP packets received : 471
-- Total IPv4 packets received : 602
-- Total IPv6 packets received : 118
-- Total Other link packets received : 136
-- Total IPinIPv4 packets received : 0
-- Total IPinIPv6 packets received : 0
-- Total GRE packets received : 0
-- Total TCP packets received : 529
-- Total UDP packets received : 28
-- Total ICMP packets received : 31
-- Total Other transport packets received : 132
--
-- Total sessions tracked : 62
-- Total assets detected : 12
-- Total TCP OS fingerprints detected : 2
-- Total UDP OS fingerprints detected : 0
-- Total ICMP OS fingerprints detected : 0
-- Total DHCP OS fingerprints detected : 0
-- Total TCP service assets detected : 0
-- Total TCP client assets detected : 0
-- Total UDP service assets detected : 0
-- Total UDP client assets detected : 0
-- libpcap:
-- Total packets received : 1393
-- Total packets dropped : 0
-- Total packets dropped by Interface : 0
[*] prads ended.
root@gateway1:/home/group1#
```

This output is the terminal output we get by running PRADS on the specified network. This output is just a statistical information about what assets have been detected and their number, total packets and types of OS finger prints detected. But the log file is by default logged into '*/var/log/prads-asset.log*'. Actually it is in the log file the detail of the assets is found where every entry in the logfile is put in one line and one line contains eight fields describing an asset. The eight fields are namely *asset,vlan,port,proto,service,[service-info],distance,discovered* in which the asset is represented by IP address on the machine in charge. In order to have a good clue of the suggested visualization ways, it is necessary to look into the log file of PRADS. The log file is written in

an eight column format of one line in a row and a sample output looks as follows.

Listing 4.2: Sample PRADS Logfile Output

```
asset,vlan,port,proto,service,[service−info],distance,discovered
10.0.0.1,0,58078,6,SYN,[S10:64:1:60:M1460,S,T,N,W4:.:unknown:unknown:\
link:ethernet/modem:uptime:1554hrs],0,1399297249
10.0.0.2,0,22,6,SYNACK,[5792:64:1:60:M1460,S,T,N,W5:ZAT:Linux:2.6 \
(newer, 5):link:ethernet/modem:uptime:1554hrs],0,1399297249
10.0.0.2,0,22,6,SERVER,[ssh:OpenSSH 5.5p1 (Protocol 2.0)],0,1399297249
10.0.0.1,0,22,6,CLIENT,[ssh:OpenSSH 5.9p1 (Protocol 2.0)],0,1399297249
10.0.0.4,0,22,6,SYNACK,[5792:64:1:60:M1460,S,T,N,W1:ZAT:Linux:2.6 \
(newer, 1):link:ethernet/modem:uptime:1554hrs],0,1399297309
10.0.0.4,0,22,6,SERVER,[ssh:OpenSSH 5.5p1 (Protocol 2.0)],0,1399297309
10.0.1.2,0,22,6,SYNACK,[5792:63:1:60:M1460,S,T,N,W1:ZAT:Linux:2.6 \
(newer, 1):link:ethernet/modem:uptime:1554hrs],1,1399297498
10.0.1.2,0,22,6,SERVER,[ssh:OpenSSH 5.5p1 (Protocol 2.0)],1,1399297498
10.0.1.2,0,53,17,CLIENT,[unknown:@domain],1,1399297500
128.39.89.8,0,53,17,SERVER,[unknown:@domain],2,1399297500
128.39.89.8,0,53,17,SERVER,[domain:DNS SQR No Error],2,1399297526
10.0.1.2,0,46894,6,SYN,[S4:63:1:60:M1460,S,T,N,W1:.:Linux:2.6, seldom 2.4 \
(older, 2):link:ethernet/modem:uptime:1554hrs],1,1399297526
173.252.110.27,0,80,6,SYNACK,[14480:82:1:60:M1460,S,T,N,W8:ZAT:unknown:unknown\
:link:ethernet/modem:uptime:954hrs],46,1399297526
10.0.1.2,0,80,6,CLIENT,[http:Wget/1.12 (linux (gnu))],1,1399297526
173.252.110.27,0,80,6,SERVER,[unknown:@www],46,1399297527
```

4.2.1 Bar Chart Visualization

This visualization mechanism implements the visualization tools microsoft excel and NodeXL for the visual presentation of the data obtained from the terminal window. The data used in this subsection is not obtained from PRADS log file but it is collected from the terminal window shown when PRADS ends its sniffing process properly.

Figure 4.3 represents the summary of different asset types detected by PRADS in the network. The figure 4.3 below shows the UDP-client,UDP-service,TCP-client,TCP-service assets as well as the total number of assets detected. PRADS reports all these reports in textual format but it gives better analysis if it is visualized in charts and graphs.

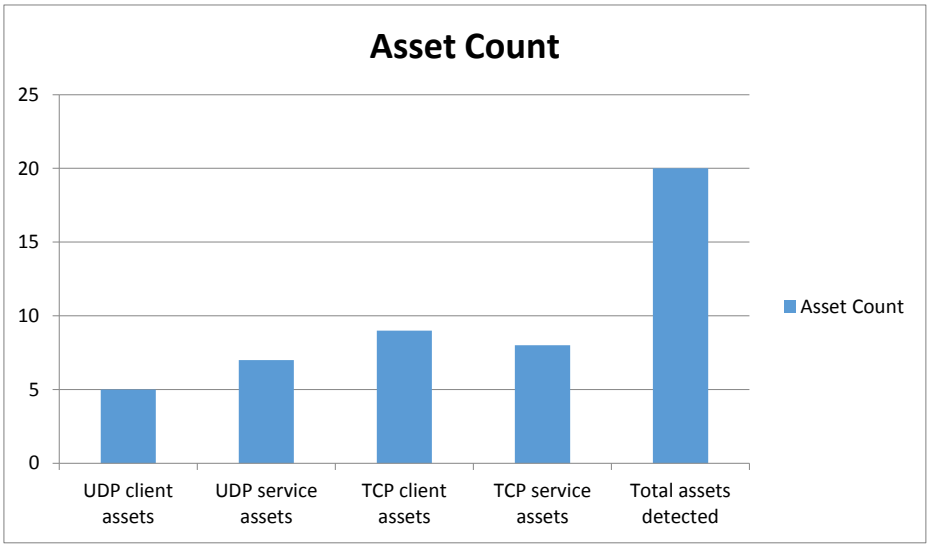

Figure 4.3: Asset Count

Similarly the visualization way presented in figure 4.4 show the count of the client and server services being served in the network with in different time intervals of the day. It will report to the network analyst in which time of day more client is being served and in which interval more server-service is being served.

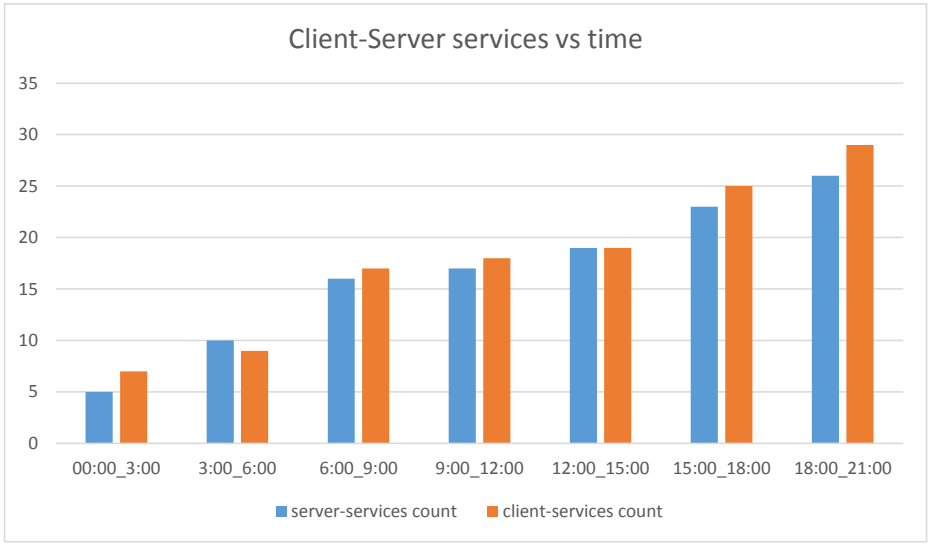

Figure 4.4: Client and Server services vs time

The visualization in figure 4.5 shows the count of the ports participating in the network traffic at each time interval. The ports are represented with different colours, the vertical line shows the count of the respective ports and the horizontal line shows the time interval in hours. From this figure it can be concluded which port is mostly used at which time interval.

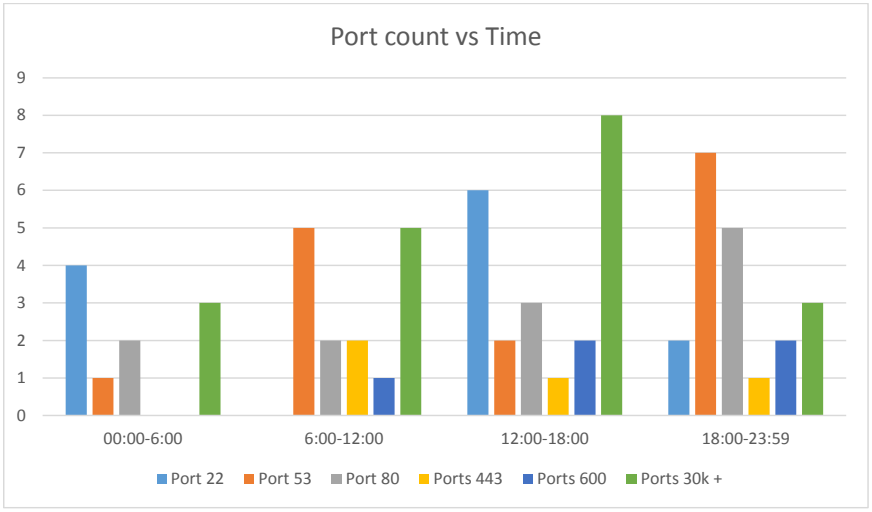

Figure 4.5: Ports Count vs Time

Finally figure 4.6 shows the age of services that can be visualized in a graphical way by looking into the time slots the services have been served. That is the time of connection to the ports in charge of the respective services. In this figure's case two services, SSH and HTTP services, can appear at different time intervals of a daily or weekly report and the age will be represented by different fading colors. The visualization in this figure shows IP1 ran SSH-client and HTTP-server earlier than IP2. That's why IP1 is represented by the fading colour. This idea can be represented as follows in figure 4.6.

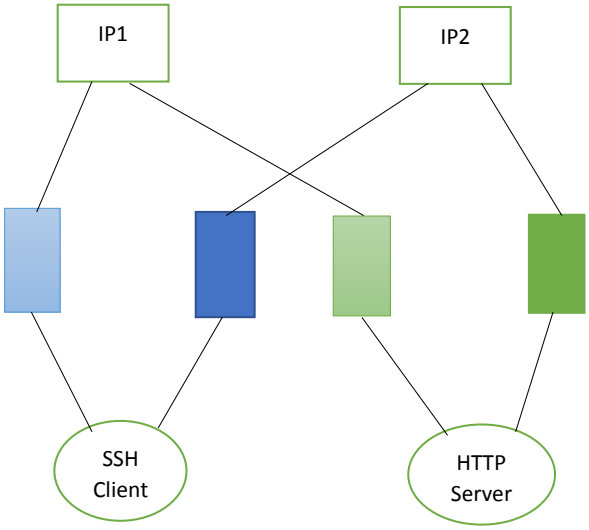

Figure 4.6: Age of Services

4.2.2 Visualization in Tree Maps

Visualizing in Tree Maps is one way of visualizing network data suggested in this thesis. Visualization presented in figure 4.7 shows how tree maps can be used to represent protocol-service mappings. The map shows the detail of the different protocols and respective services received by the sniffing tool out of the total packets received. It is also possible to visualize the amount of each packet received to show the ratio of the service types being served in the network.

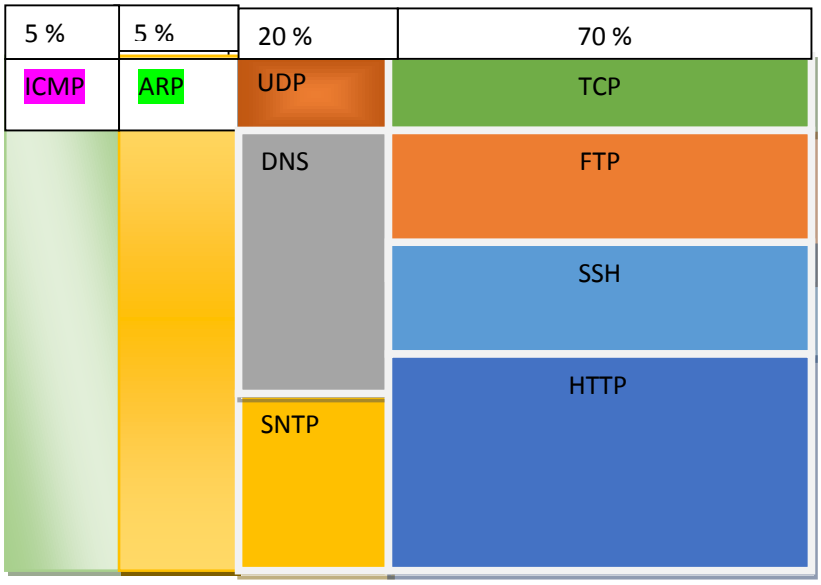

Figure 4.7: Protocol-Service Mapping

The visualization way demonstrated in figure 4.8 is also another tree map visualization method where clustering is used . First of all the IPs are clustered as internal and external IPs. Then the internal IPs will be clustered subnet wise as the following figure 4.8. By clicking the specific subnets in the tree map, the IPs belonging to each subnet will be displayed and further clicking will lead to the services detected by each individual asset. Similarly those belonging to the external IP will be displayed by clicking it or zooming into it.

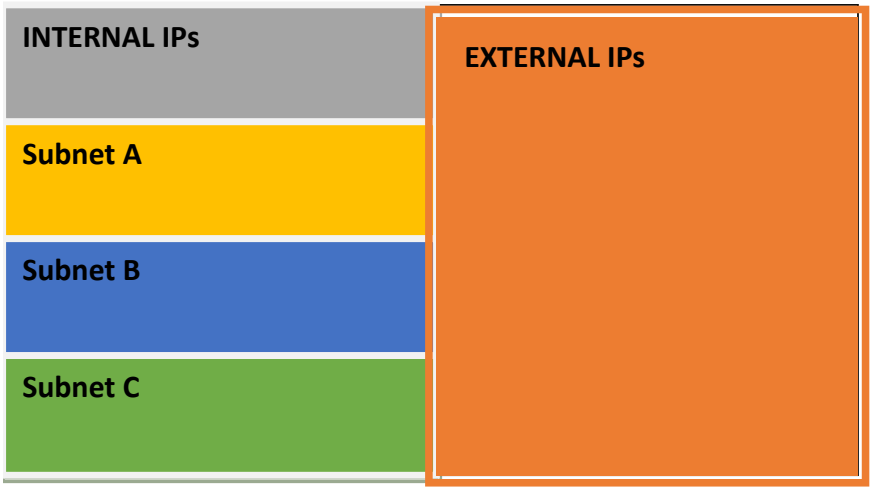

Figure 4.8: Clustering in Subnets

Figure 4.9 implements tree map visualization mechanism and shows the count of occurrence of a certain IP-port mapping. There are different shapes and colours for each port. In the case of port 22 the larger circle shows more number of appearances and the smaller circle shows less number of appearances. By summarizing the occurrence of a specific IP-port mapping in this figure's case, the magnitude of the IP-port connections can be summarized using different shapes and sizes as shown in the figure below . Similar visualization can be made using IP-service mapping where the occurrence is counted by the mapping between the assets and the services they run during the detection time. This can be represented by how many times an asset acted as a server and how many times it acted as a client at the specific time interval since there is a situation where a single asset can act both as server and client.

IP	Port	Count of appearance
IP1	22	10
IP2	22	30
IP3	80	30
IP4	53	8

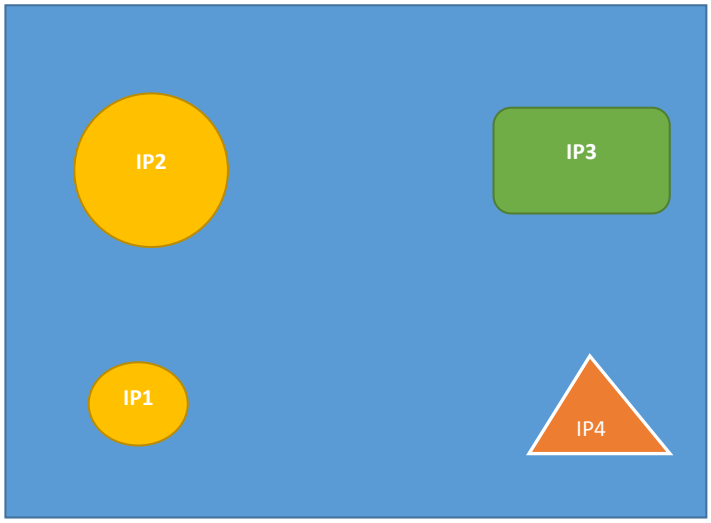

Figure 4.9: IP-Port Mapping Appearance Count

4.2.3 Internal and External IP Mapping

The following figure's visualization way also implements clustering for the external IP addresses and shows internal IP and port mapping. It shows mapping of clustered external IP with the internal IPs and the ports associated with them. External IPs are clustered to one node if the emphasis is going to be given for the internal IP addresses. The figure 4.10 in general shows the incoming traffic where the source node is clustered into one node as External, and the event nodes are the internal IP addresses and the target nodes are the ports.

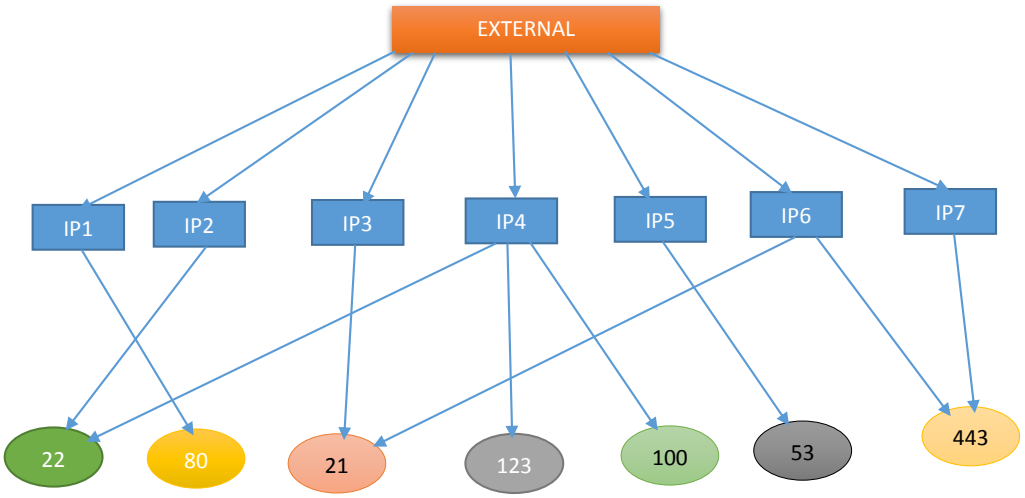

Figure 4.10: Clustering and IP-Port Mapping

In the other method of visualization demonstrated in figure 4.11, The internal IPs will be mapped with their respective ports on one side and the external IPs on the other side. Further more the external and internal IPs will be mapped by the services associated with each IP. This is the case where there is network traffic flow with source IP: source port and destination IP: destination port. The services being served in this case are associated with the ports. This source destination mapping can be represented as in the following figure 4.11 [53].

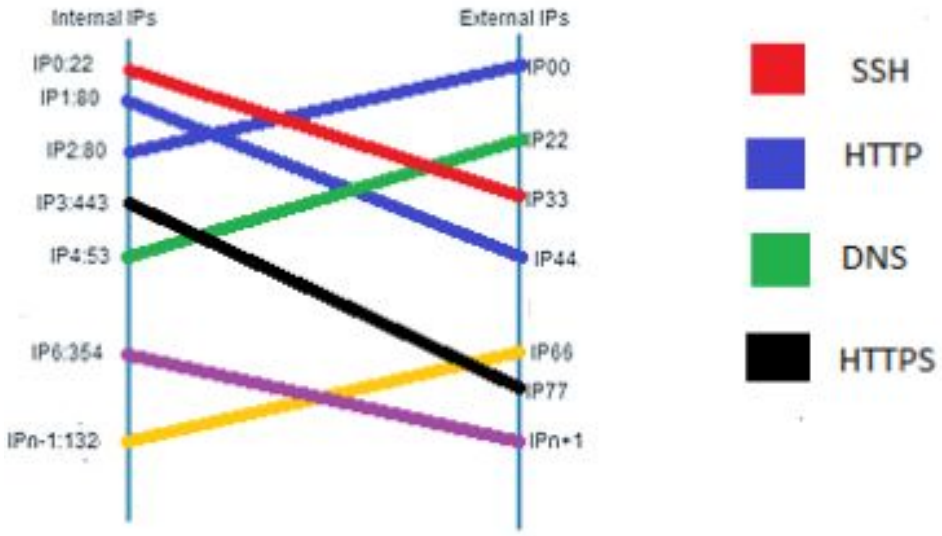

Figure 4.11: Internal-External IPs Clustering By Mapping with Ports and Services

4.2.4 Visualization Using Asset-Port-Service Mapping

The figure 4.12 in this sub section shows how visualization is possible using asset and service mapping. The central nodes are the asset IPs running different services. In such a situation,one asset can be both client and server as shown in the figure below. An asset can be a client and at the same time can give services. In the figure the IP3 is supposed to represent external address where as IP1 and IP2 are internal IPs. A summarized graphical presentation can be represented as in the figure 4.12 bellow.

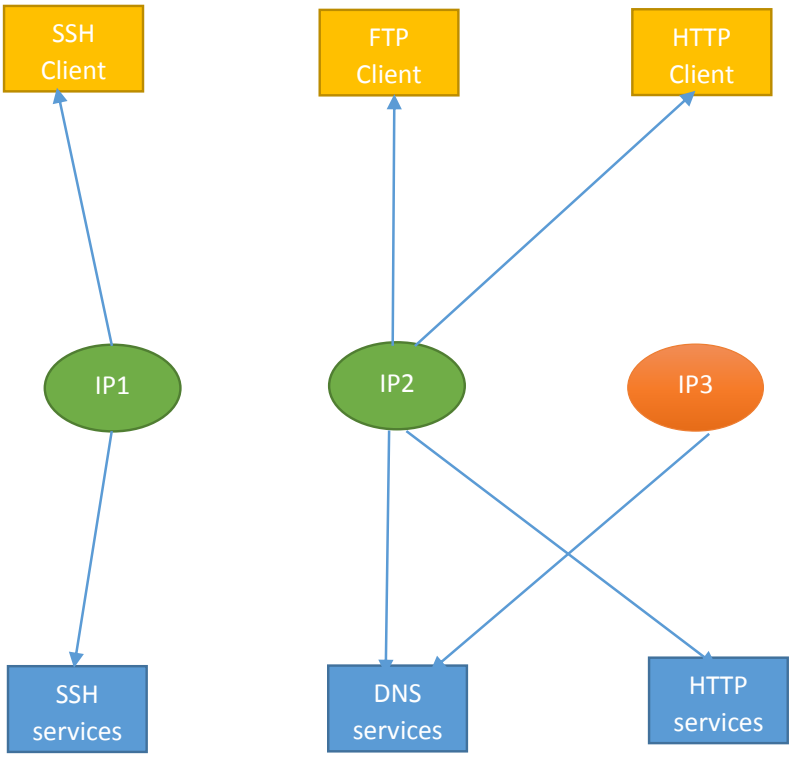

Figure 4.12: IP-Service Mapping

Similarly the following figure 4.13 shows the IP-port-service mapping. Where we have a clear distinction between source nodes which are the IP addresses involved, the event nodes which are the ports and finally the target nodes associated with the ports displayed in the event nodes. The target nodes are specified as server and clients. We call them services because they belong to the column services in the PRADS output format. This figures representation is similar to the above presentation in figure 4.12 except that there is some rearrangement in the nodes. In the figure 4.13 some nodes are acting as both server and client which is also true in the actual output of PRADS. As shown here IP3 can give webservices and at the same time it can run a web-browser

acting like a client. Such a scenario can be represented in the figure 4.13.

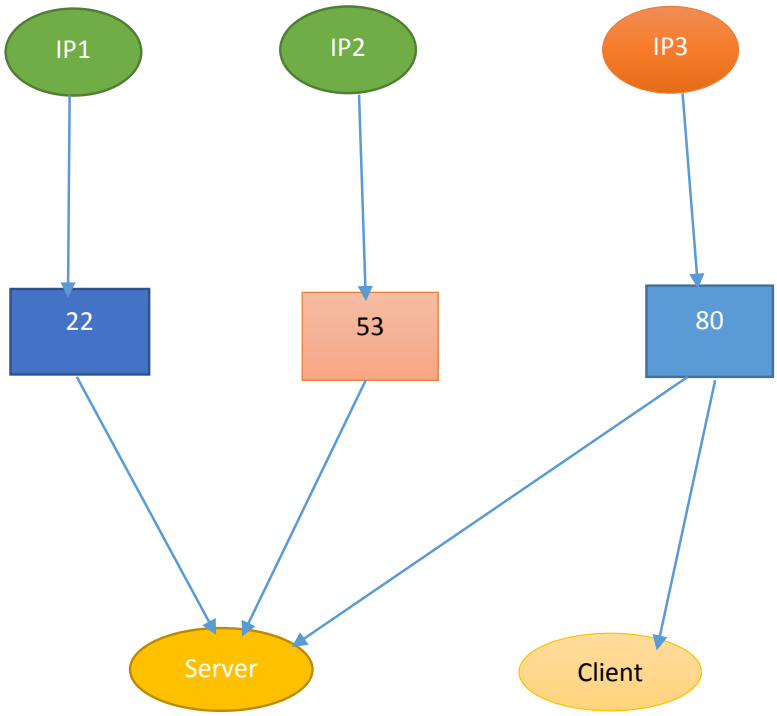

Figure 4.13: Source-Event-Target Mapping

The visual presentation shown in figure 4.14 is a solution to tackle the problem created due to the increased number of mappings. One scenario that can decrease the complexity created due to more number of nodes and mapping is edge-colouring. As the edges go from one node to another, as the number of nodes increases it could be difficult to identify which edge is coming from which node and which edge is going to which node. This situation is created as we have 3-level nodes called source,event and target, where the event node is the one that connects the source and target nodes. Therefore by colouring the edges, it will be easier to identify the asset-service mappings. It will be easier to show which host is acting as server at which port, and which asset is a client at which port. Such scenario is shown in the following figure.

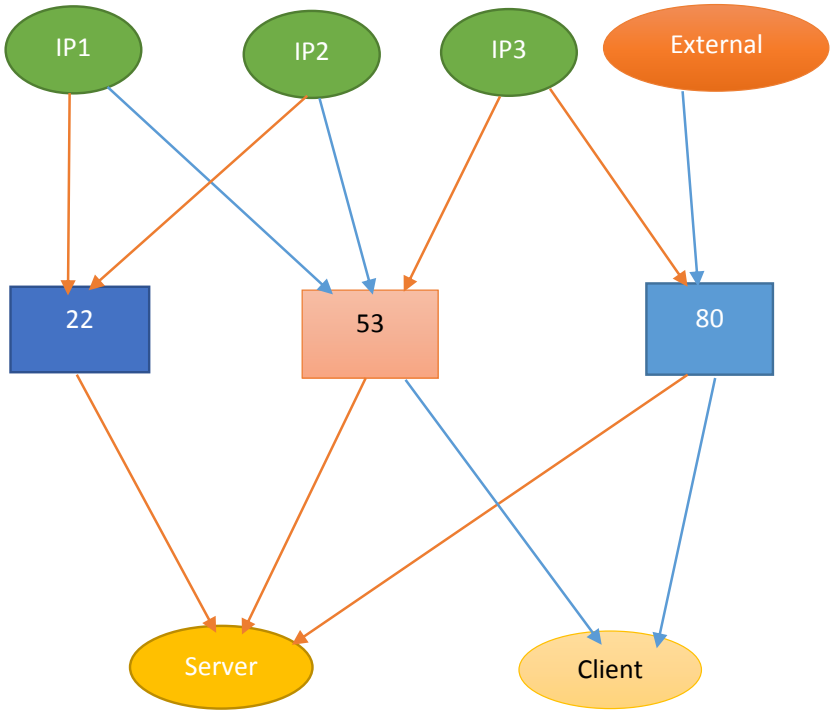

Figure 4.14: Node-Edge Based Customization

Part III

4.3 Proof of Concept

In this section the realization of the prototype will be demonstrated by automating the selected open-source visualization tools,afterglow and graphviz. As mentioned in the earlier chapters, the tools are mainly chosen according to their simplicity and usability to visualize the PRADS output data and their suitability for automation. Therefore the proof of concept will be demonstrated using these visualization tools and the prototype developed to incorporate the tools with the log files. The prototype is developed using a perl scripting language as the visualization tools selected are also perl based versions.

4.3.1 Prototype Testing

The prototype is developed in a number of phases. It is developed in a controlled environment where every network traffic is generated manually. At the beginning, the prototype development started by parsing all the IP addresses from the asset column collected by running PRADS in one of the interfaces of the gateway in the network setup shown in the methodology chapter. Then the parsing is extended to more columns of the log file where we would like to visualize using the mentioned tools.

Similarly the visualization tools were first installed and configured in our gateway. The tools installed and configured are afterglow and graphviz. The tools are tested manually using some rows of the PRADS output data. Some parsed data was piped from two rows of the output data and fed to the afterglow which gave a graphical language and this graphical language was further used by graphviz to give visual outputs. Every thing started from scratch to study how each tool implements the visualization process. Some customizations like colouring was also implemented at the beginning.

After making some manual tests, PRADS was made to run in the virtual network that is assigned to this project. PRADS was made to collect data in a controlled way by creating different network traffic from the machines in our virtual network. Some scripts and linux commands were run from inside the machines in the internal network to create more traffic in order for PRADS to collect it. Some new services like apache were installed in one of the internal machines in the internal network to check if PRADS can detect every newly installed service as well. In fact PRADS was able to detect everything that is running in the network; in which the detection was done passively.

Finally , more inclusive and general script was developed to integrate all the

steps involved in visualizing the log files obtained from PRADS. The script is automated to make data collection in a daily basis making PRADS to log into a file name with year/month/day-prads.csv format. For example the output file can be put in the format 20140505-prads.csv. This output file is given to the parsing script, *parser.pl* which pipes the required fields that afterglow needs to visualize. The afterglow takes the parsed CSV files and creates graphical languages. Afterglow has an option to consider during visualization process, one of the options is the properties file(configuration file) which helps a lot in customizing the graph output. The customization includes assigning different colours to nodes and edges; clustering similar nodes; and assigning different shapes and sizes to nodes. Afterglow has more options and will be discussed in the next sections when demonstrating the proof of concept using this prototype.

The visualizing script collects the visualization information from the Afterglow and again pipes it to the Graphviz visualization tool. Graphviz uses different graph layouts such as *dot, neato, circo* and *twopi* to present the graphs visually for further analysis. Therefore the visualization tools afterglow and Graphviz are working together in this prototype to give us sensible visualization results that save time and effort for network and system administrators to understand their network.

After step by step development of the prototype, it was first tested in the local network shown in section in figure 3.1, the experiment environment setup, by running PRADS in the internal interface eth1 of the gateway gateway1. As a result, visual output was obtained and different customizations such as clustering; assigning nodes different colours,shapes, and size were practised to give the best possible output. Traffic was generated by all means including using scripts and linux commands from the internal machines as much as possible to get more data to visualize. Services were stopped,started,installed, and removed from assets of the internal network to identify if PRADS does detect the services which in our case PRADS did detect the status of the services in most scenarios.

Checking all the scenarios in the internal network, everything went right and output graphs were customized and collected. After testing the script in the internal network by running PRADS on eth1 and visualizing the output, then PRADS was made to run on eth0 where the internal network is considered to be the network 128.39.120.0/24. Consequently output was rendered and different customizations were practised the same way it was done in the internal network. The graphs were made to be presented in different layouts in order to choose the best way to present the graphs to users and network analysts.

Similarly the testing was extended to the network of HIOA by running PRADS in the HIOA network and collecting data to be visualized using the same script that was used in the internal network. The same script is used in all the scenarios that were tested but the one changing is the configuration file or the

properties file which is customized according to the volume of the traffic where PRADS is running. As the volume of the logfile increases it is important to filter out those that are not important for our consideration. Otherwise the graph can be too much crowded. It can be sometimes necessary to make minimum requirements for a node to appear in the graph based on the number of occurrences of a node in the traffic usually applied to limit number of source nodes. There can be a limit for a node to be shown based on the number of connections it has made also which can be usually applied on event nodes.

The results are obtained by making as many tests as possible in this way. Now is the time to display the results collected according to our prototype.

4.3.2 Visualization from Running PRADS in Eth1

When PRADS is run in the internal interface eth1 of our network shown in the methodology chapter, the networks 10.0.0.0/24, 10.0.1.0/24, and 10.0.2.0/24 are considered as internal networks and the rest are treated as external networks in this subsection of visualization demonstration. The prototype was run by specifying PRADS to sniff on **eth1** using a perl script language, **visualizer.pl**. The script was run as follows automating from data collection to visual presentation.

Listing 4.3: Running The Visualizer Script

```
1  perl parser.pl -f $output_file | perl /home/group1/afterglow/afterglow-master \
2  \afterglow.pl -c /home/group1/afterglow/afterglow-master/color.properties |\
3     dot -Tgif -o /home/group1/afterglow/afterglow-master/$output_afglow.gif
```

The functioning of this script is described in the prototype testing subsection in the methodology chapter. The visualizer script that contains the above command is common for all visualization demonstrations. The different changes such as customization and filtering are made in the **properties** file. Another difference could be the layout in the graphviz which is sometimes dot, neato,circo or twopi. After the *properties* file we can use as many command line arguments of afterglow as possible. The command line arguments are node limits, edge length, node count, etc as described in section 5.1.

Running the **visualizer.pl** script by making PRADS listen to eth1 network traffic resulted in the following visual presentations. The visual presentations will be presented along with the property files used to customize the graphs since without the property files the graphs will be boring and difficult to analyze. Therefore property file sometimes called configuration file description is as important as the graph itself. The reason for this is users and network analysts cannot understand the graph presentations as it is not possible to add legends in the graphical presentations using a script. But a legend can be added manually as shown here in some of the figures. There is no much traffic involved

when PRADS is run in eth1; the network is controlled and traffic is deliberately generated from the virtual machines using different linux commands and scripts. Therefore the visualizations shown here are mostly using the dot layout format of graphviz for final graph presentations.

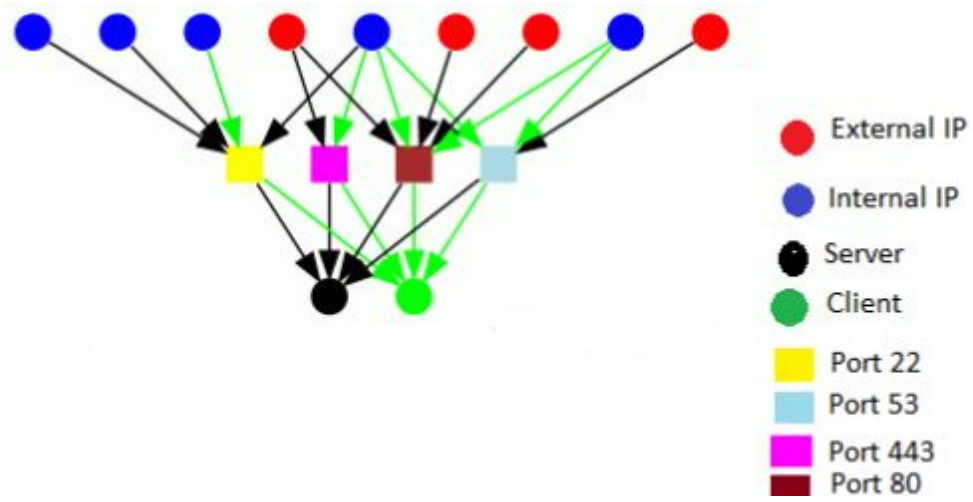

Figure 4.15: Emphasis on Common Ports Using Dot Layout

The figure 4.15 shows the network mapping between the assets, ports, and services. On this visualization emphasis is given on the common ports 22, 53, 80, 21 and 443 and are assigned different colours each and other ports are filtered out.This graph's visualization uses the **dot** layout. We have different assets participating in the network and the assets are classified as external and internal IPs by assigning red colour to external IPs and the internal ones are assigned blue colour throughout the whole visualization process in this subsection section. To avoid confusion edge colouring is used here; green lines lead to clients and black lines lead to servers. The *service* field in the PRADS output contains TCP-flags such as SYN, FIN, RST , etc. But these TCP-flags are filtered out in our visualization since they are not considered important in our visualization demonstration. The customization and filtering for figure 4.15 is done in the properties file as follows.

Listing 4.4: Properties File

```
1  color.source="blue" if ($fields[0]=~/^10\.0\..*/)
2  color.source="red"
3  color.event="yellow" if ($fields[1] eq "22")
4  color.event="lightblue" if ($fields[1] eq "53")
5  color.event="brown" if ($fields[1] eq "80")
6  color.event="pink" if ($fields[1] eq "21")
7  color.event="magenta" if ($fields[1] eq "443")
8  color.event="invisible"
9  color.target="black" if ($fields[2]=~/^SERVER/)
10 color.target="green" if ($fields[2]=~/^CLIENT/)
11 color.target="invisible"
12 color.edge="black" if ($fields[2]=~/^SERVER/)
```

```
13 color.edge="green" if ($fields[2]=~/^CLIENT/)
14 color.edge="invisible"
15 size.edge=1;
```

Using the same property files but different layout called the neato layout we have a 3-dimensional graph shown in the figure 4.16. Everything is the same as the visualization in figure 4.15 except the layout. The neato layout is usually used when there is large amount of data to be visualized to avoid cluttering or congestion. To partially show the layout part of the script, the dot will be replaced by neato in the script system command.

```
| dot -Tgif -o /home/group1/afterglow/afterglow-master/$output_afglow.gif
becomes
| neato -Tgif -o /home/group1/afterglow/afterglow-master/$output_afglow.gif
```

This execution presents the graph shown here in figure 4.16

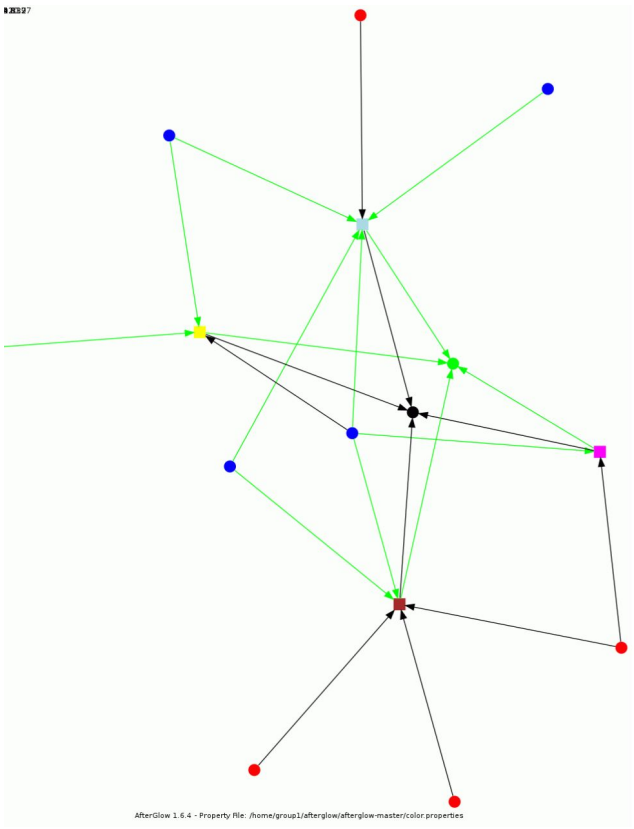

Figure 4.16: Emphasis on Common Ports Using Neato Layout

The next visualization output shown in figure 4.17 implements the clustering of similar nodes aimed at decreasing congestion of nodes.Such visualization way is usually important when we want to have a general overview of our network and when we have large number of nodes. In figure 4.17 all the external nodes are clustered to one node and the ports less than 1024 are clustered to the be same colour because mostly the services are served on ports less than 1024. The clustered visualization output is shown as follows in figure 4.17 In order to have

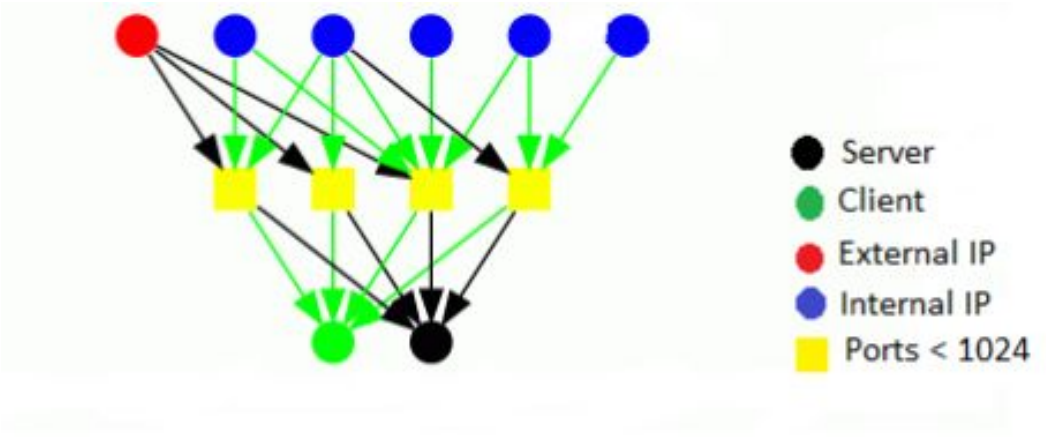

Figure 4.17: Clustering of similar Nodes

a visualization like in figure 4.17 the property file is written as follows :

Listing 4.5: Clustering Properties file

```
1  cluster.source="External" if ($fields[0]!~/^10\.0\..*/)
2  color.source="red" if (field() eq "External")
3  color.source="blue"
4  color.event="yellow" if ($fields[1]<1024)
5  color.event="invisible"
6  color.target="black" if ($fields[2]=~/^SERVER/)
7  color.target="green" if ($fields[2]=~/^CLIENT/)
8  color.target="invisible"
9  color.edge="black" if ($fields[2]=~/^SERVER/)
10 color.edge="green" if ($fields[2]=~/^CLIENT/)
11 color.edge="invisible"
```

Another visualization mechanism shown in figure 4.18 is node customization based on the number of occurrences of the nodes. The size of a node increases when the number of occurrence increases. There is a fixed size of a node specified on the properties file. It is like when a node occurs n times on the network , the size will be n times the fixed size specified. The visualization looks as in the figure 4.18

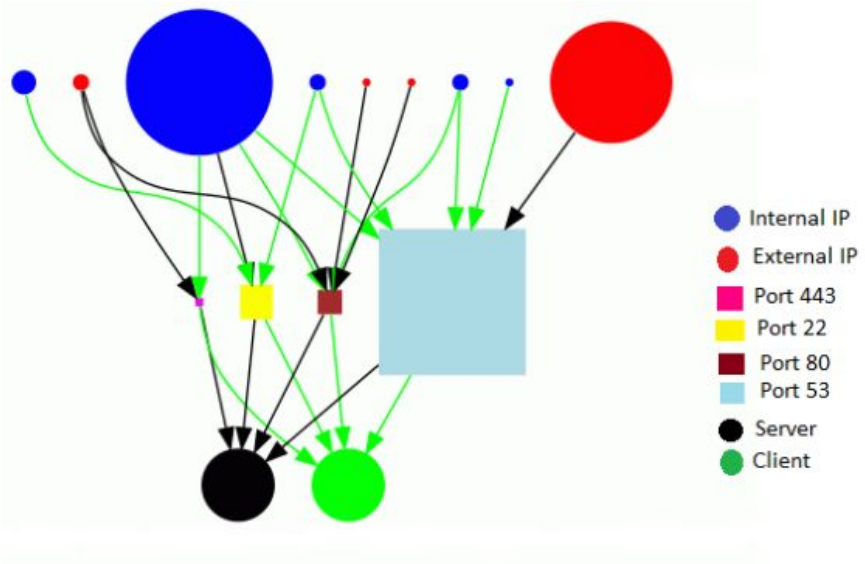

Figure 4.18: Node Count Consideration

In order to have a visualization like in the figure 4.18 the properties file is written in the following way.

Listing 4.6: Node-size Customization Properties

```
color.source="blue" if ($fields[0]=~/^10\.0\..*/)
color.source="red"
color.event="yellow" if ($fields[1] eq "22")
color.event="lightblue" if ($fields[1] eq "53")
color.event="brown" if ($fields[1] eq "80")
color.event="pink" if ($fields[1] eq "21")
color.event="magenta" if ($fields[1] eq "443")
color.event="invisible"
color.target="black" if ($fields[2]=~/^SERVER/)
color.target="green" if ($fields[2]=~/^CLIENT/)
color.target="invisible"
color.edge="black" if ($fields[2]=~/^SERVER/)
color.edge="green" if ($fields[2]=~/^CLIENT/)
color.edge="invisible"
size.source=$sourceCount{$sourceName};
size.event=$eventCount{$eventName};
size=0.5
```

4.3.3 Visualization from Running PRADS in Eth0

When PRADS is run in the external interface Eth0 of our network lab setup, data is collected in a similar way as in the internal interface Eth1. Everything is the same except that the network 128.39.120.0/24 will be considered as our internal network in addition to the networks 10.0.0.0/24, 10.0.1.0/24, and 10.0.2.0/24. Therefore the network traffic in this case is larger and more customization and filtering is required in this case. As the number of nodes grow it becomes difficult to apply the dot layout of Graphviz; which makes the implementation of other Graphviz layouts like neato very important . In visualizations where there is large amount of log file, neato layout becomes more sensible and suitable to use. This can be demonstrated in the following figures and our eye can witness which visualization way is better and more understandable. The same amount of nodes and customizations like colouring and sizing are used; the difference is in graph layouts. Note that black edges are pointing to servers and green edges are pointing to clients throughout all visualizations in this subsection.

Sometimes when we are using the dot layout for large amount of data that is in thousands of log files, we might get processor error messages. That's why customizations like clustering are important for such scenarios. Message of this kind may not be observed when using the neato layout. The visualization shown in figure 4.19 was tested for visualization using the dot layout but the graph was difficult to adjust and was cluttered graph. But everything went right when using the neato layout as shown here in the figure 4.19.

The properties file used to show the visualization in figure 4.19 can be shown as follows.

Listing 4.7: Common Ports Properties File

```
color.source="blue" if ($fields[0]=~/^10\.0\..*/)
color.source="blue" if ($fields[0]=~/^128\.39\..*/)
color.source="red"
color.event="yellow" if ($fields[1] eq "22")
color.event="lightblue" if ($fields[1] eq "53")
color.event="brown" if ($fields[1] eq "80")
color.event="pink" if ($fields[1] eq "21")
color.event="magenta" if ($fields[1] eq "443")
color.event="invisible"
color.target="black" if ($fields[2]=~/^SERVER/)
color.target="green" if ($fields[2]=~/^CLIENT/)
color.target="invisible"
color.edge="black" if ($fields[2]=~/^SERVER/)
color.edge="green" if ($fields[2]=~/^CLIENT/)
color.edge="invisible"
size.edge=1;
```

The visual presentation shown in figure 4.19 is used to visualize a log file with 3000 lines. Such many lines of textual log file is difficult to present using a dot layout without using clustering and filtering of nodes. Similarly with the same property file but only clustering the event nodes (ports) we can have a visual-

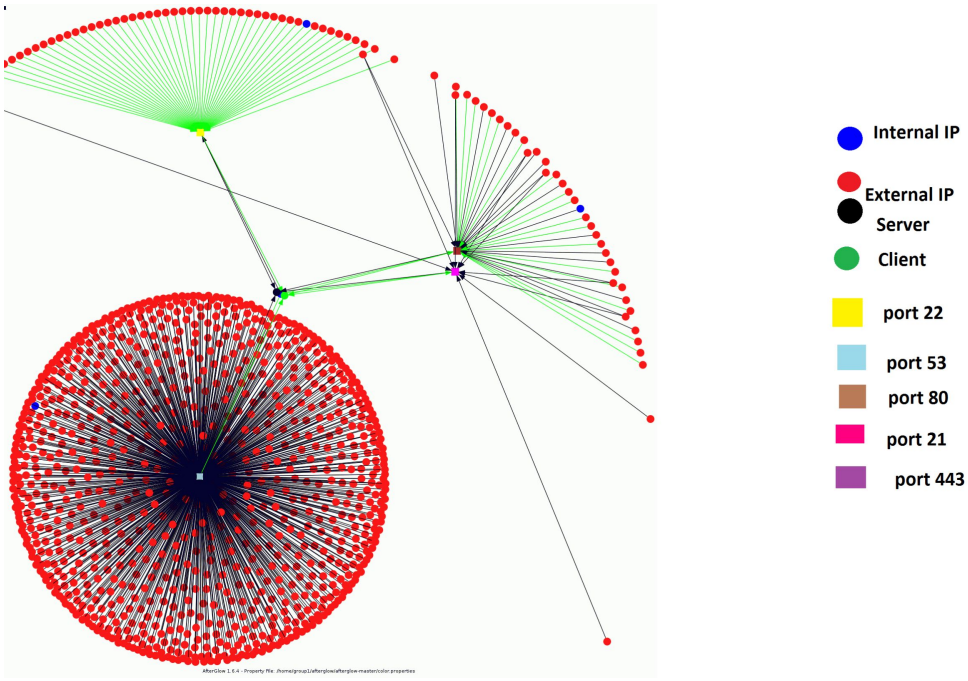

Figure 4.19: Neato Layout During Large logfile Visualization

ization using neato layout as shown in figure 4.20. It is similar presentation to figure 4.19, except the port clustering. The modified line is as follows :

```
color.event="yellow" if ($fields[1] <1024)
color.event="invisible"
```

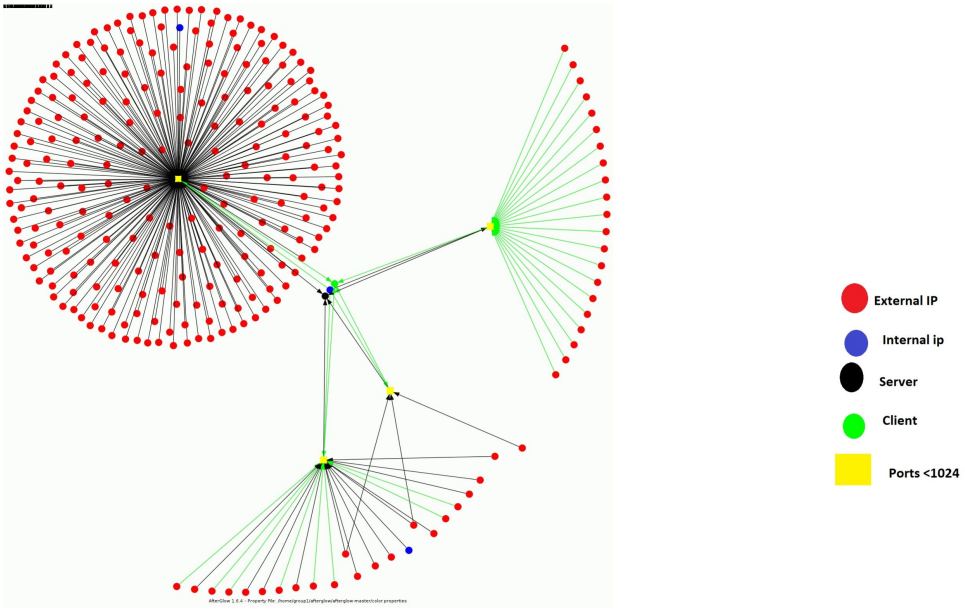

Figure 4.20: Neato Layout Using Ports Clustering

If we have to use dot layout for the same amount of log file data as in figure 4.19, then we can present the visualization as shown in figure 4.21 using different customization methods. The customizations involve clustering of similar nodes, filtering out unimportant nodes and assigning different colours and shapes to nodes and includes colouring edges differently as well. This way a better graph will be presented as shown in the figure 4.21 without any problem.

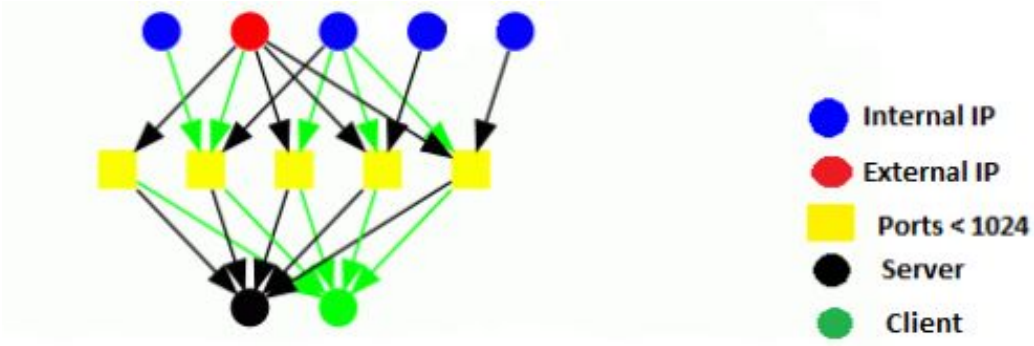

Figure 4.21: Clustering Similar Nodes

In figure 4.21 all external IPs are clustered to a single node and ports ⟨1024 are represented by one colour. The properties file used to create such visual graph is

Listing 4.8: Clustering on Eth0 Data

```
1  cluster.source ="External" if ($fields[0]!~/^128\.39\..*/)
2  color.source="red" if (field() eq "External")
3  color.source="blue"
4  color.event="yellow" if ($fields[1]<1024)
5  color.event="invisible"
6  color.target="black" if ($fields[2]=~/^SERVER/)
7  color.target="green" if ($fields[2]=~/^CLIENT/)
8  color.target="invisible"
9  color.edge="black" if ($fields[2]=~/^SERVER/)
10 color.edge="green" if ($fields[2]=~/^CLIENT/)
11 color.edge="invisible"
12 size.edge=1;
```

4.3.4 Visualization of HiOA Network

Data was collected by running PRADS at the gateway of HiOA network. The amount of data collected was huge estimated around 4500 lines (rows) containing both IPv4 and IPv6 addresses. The visualization process for this collected data is made the same way as the other visualization demonstrated so far. The IP addresses can be classified as internal and external IPs. The data collected is too big to be visualized using the dot layout of Graphviz with out filtering unimportant nodes. Therefore the neato layout will be implemented by Graphviz to present the data in graphs. The visualization is shown in figure 4.22 giving emphasis on the common ports 21, 22, 53, 80 and 443.

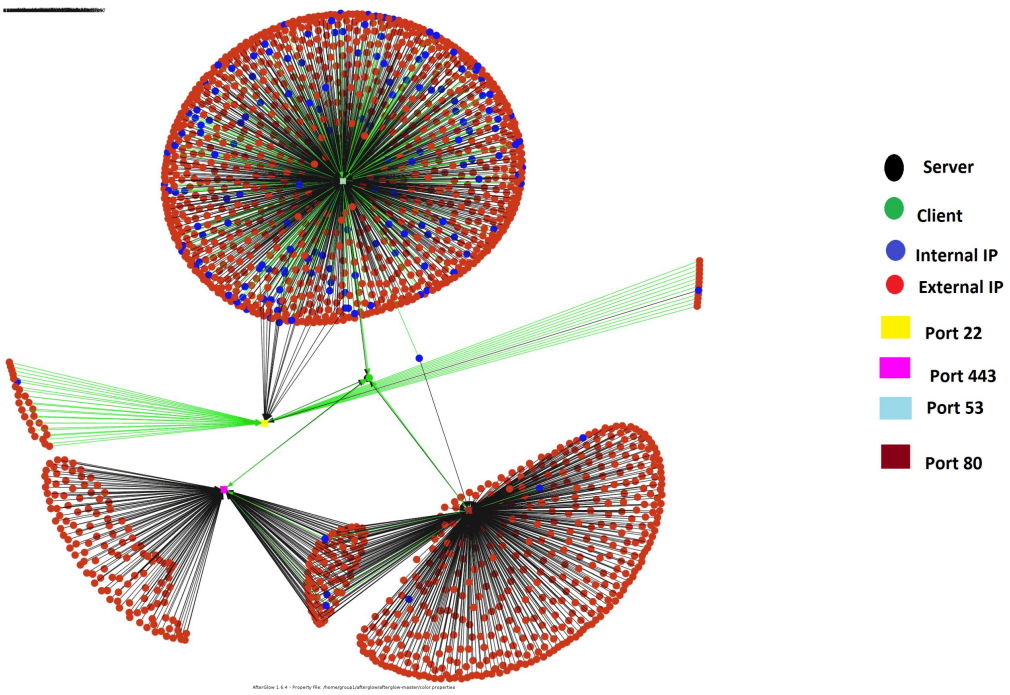

Figure 4.22: HiOA Network Data Visualization

The properties file used in order to have the visualization shown in figure 4.22 can be shown as follows :

Listing 4.9: Properties with Emphasis on Common Ports

```
color.source="blue" if ($fields[0]=~/^2001:700:700:..*/) #IPv6
color.source="blue" if ($fields[0]=~/^128\.39\..*/)
color.source="blue" if ($fields[0]=~/^10\.0\..*/)
color.source="red"
color.event="yellow" if ($fields[1] eq "22")
color.event="lightblue" if ($fields[1] eq "53")
color.event="brown" if ($fields[1] eq "80")
color.event="lightyellow" if ($fields[1] eq "21")
color.event="magenta" if ($fields[1] eq "443")
color.event="invisible"
color.target="black" if ($fields[2]=~/^SERVER/)
```

```
color.target="green" if ($fields[2]=~/^CLIENT/)
color.target="invisible"
color.edge="black" if ($fields[2]=~/^SERVER/)
color.edge="green" if ($fields[2]=~/^CLIENT/)
color.edge="invisible"
size.edge=1;
```

It is also possible to have a visualization which is more general for the ports than specifying the common ports. The reason is there could be other important ports than those specified ports such as 22, 21, 53, 80 and 443. Most services exist at ports less than 1024; hence we can give the same colour for the ports less than 1024 during our visualization. Doing this leads us to figure 4.23

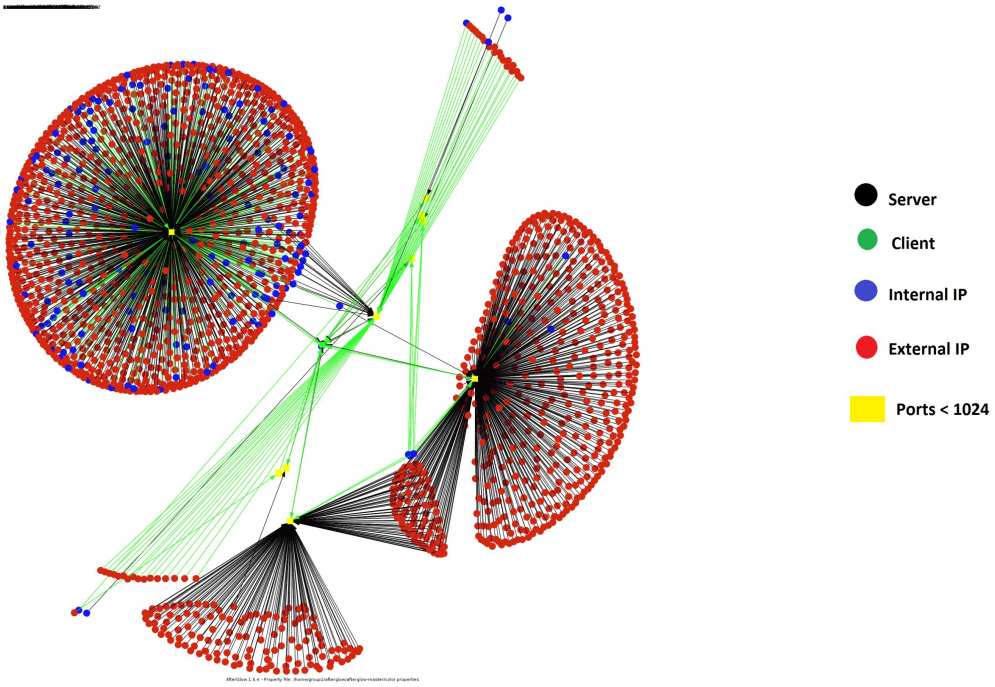

Figure 4.23: HiOA Network Data Visualization Using Clustering

In this case there are more ports than those shown in figure 4.22 which gives more eagerness for network and system administrators to know what these ports are. The figure shows additional services are running on those ports other than what we normally know about the common services such as SSH, DNS and HTTP services. Although the picture in figure 4.23 doesn't tell us the exact port or service running, we will mediately be aware that other services are running at some ports. This is one advantage of visual presentation over textual presentation of data.
The properties file for figure 4.23 is similar to that of figure 4.22. The difference is only with the event node customization. The event node is written as follows in this case.

```
color.event="yellow" if ($fields[1]<1024)
```

```
color.event="invisible"
```

In order to show how filtering out nodes works when automating the afterglow in our prototype, we can visualize only the source nodes that occurred 5 times and more in the network data collected from HiOA. The visualization shown in figure 4.23 can be further reduced to the visualization shown in figure 4.24 where a dot layout is implemented. The visualization demonstrated in the fig-

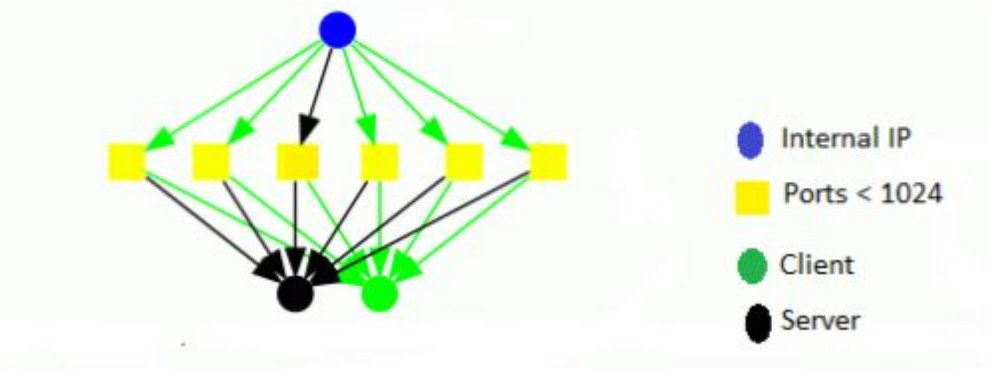

Figure 4.24: HiOA Network Data Visualization Using Filtering

ure 4.24 forces us to investigate the asset with the specified IP which is IP 128.39.89.9 in our case. We can play more with the command line arguments of Afterglow to check the frequency of the occurrence of assets by specifying the arguments f and/or g. The same properties files is used as in the case of figure 4.23 but the filtering process is done in the command line arguments of Afterglow.

4.3.5 Visualization Test on Alto-Openstack

The prototype testing was extended further to the HiOA open-stack, Alto. Data was collected by running PRADS on the open-stack and the collected data was visualized using the prototype in a similar way like the previous scenarios. The data collected was not too much and visualizing the data using dot graph layout is enough. Emphasis will be given to the common ports in this case, to know what services are running on the open-stack. To have a general idea of the SSH, HTTP, and DNS services, the properties file was specified to have a visualization in figure 4.25. The properties file used to make such visualization is listed below.

Listing 4.10: Properties on Alto Data

```
1 color.source="blue" if ($fields[0]=~/^2001:700:700:..*/)
2 color.source="blue" if ($fields[0]=~/^128\.39\..*/)
```

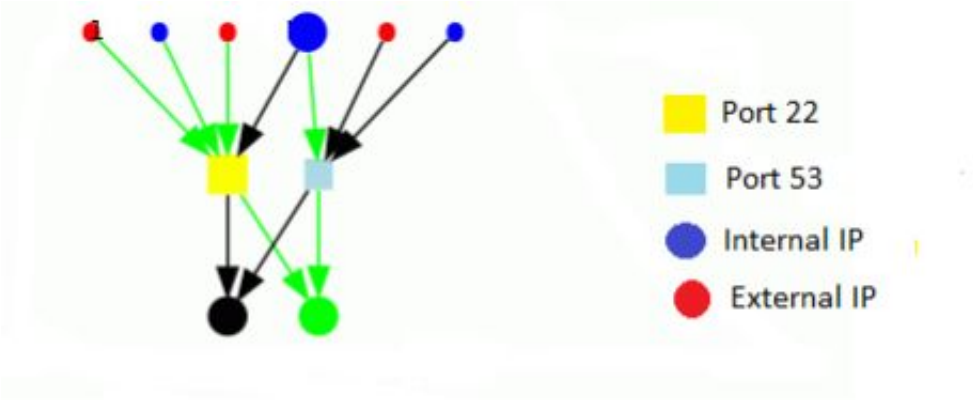

Figure 4.25: Alto-Openstack Network Data Visualization Using Filtering

```
3  color.source="blue" if ($fields[0]=~/^10\.0\..*/)
4  color.source="red"
5  color.event="yellow" if ($fields[1] eq "22")
6  color.event="lightblue" if ($fields[1] eq "53")
7  color.event="brown" if ($fields[1] eq "80")
8  color.event="lightyellow" if ($fields[1] eq "21")
9  color.event="magenta" if ($fields[1] eq "443")
10 color.event="invisible"
11 color.target="black" if ($fields[2]=~/^SERVER/)
12 color.target="green" if ($fields[2]=~/^CLIENT/)
13 color.target="invisible"
14 color.edge="black" if ($fields[2]=~/^SERVER/)
15 color.edge="green" if ($fields[2]=~/^CLIENT/)
16 color.edge="invisible"
17 size.edge=1;
18 size.source=$sourceCount{$sourceName};
19 size.event=$eventCount{$eventName};
20 size=0.5;
```

PRADS was run for about a week in the Alto-Openstack network and around 650 lines of log file was collected. It will be difficult to visualize the log file due to its size. Therefore we will be using the neato layout to visualize the data. The visualization is customized as shown in the properties file in the listing below where the source nodes are classified as internal and external using colouring and the event nodes customized to show us only the ports less than 1024. The target nodes are classified as server and client using colour customization. Such visualization is shown in the figure 4.26.

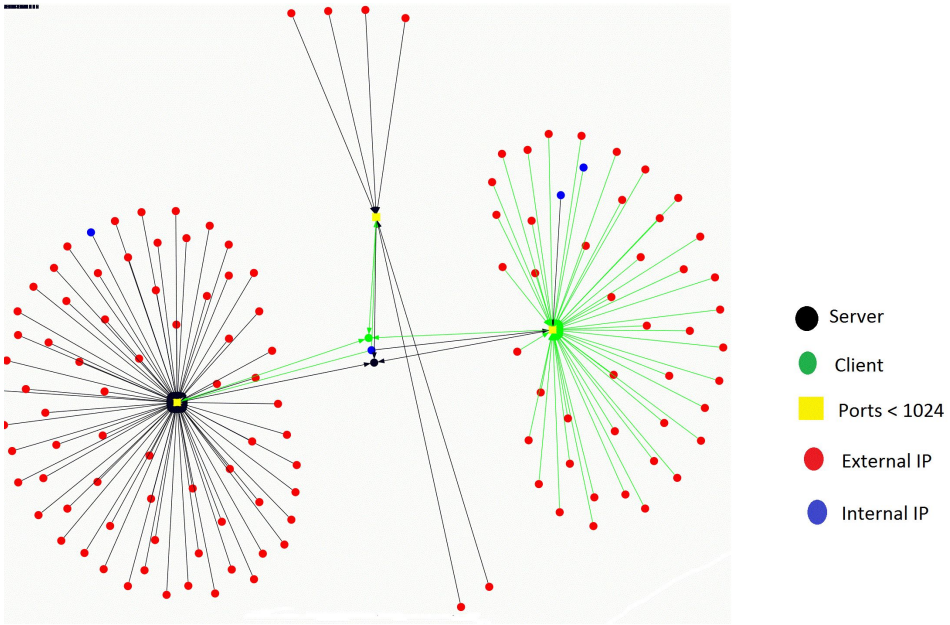

Figure 4.26: Alto-Openstack Network Data Visualization

Listing 4.11: Properties on Alto Data

```
color.source="blue" if ($fields[0]=~/^2001:700:700:..*/)
color.source="blue" if ($fields[0]=~/^128\.39\..*/)
color.source="blue" if ($fields[0]=~/^10\.0\..*/)
color.source="red"
color.event="yellow" if ($fields[1]<1024)
color.event="invisible"
color.target="black" if ($fields[2]=~/^SERVER/)
color.target="green" if ($fields[2]=~/^CLIENT/)
color.target="invisible"
color.edge="black" if ($fields[2]=~/^SERVER/)
color.edge="green" if ($fields[2]=~/^CLIENT/)
color.edge="invisible"
size.edge=1;
```

Chapter 5

Analysis

This chapter presents the evaluation of the results described in the previous chapter,analyzing the results in three parts as classified in the results chapter. The detailed analysis for each part will be given accordingly.

Part I

5.1 Open-source Data Visualization Tools

Many open-source visualization tools have been assessed and investigated.There are so many visualization tools that are used for different purposes and applications. But the tools summarized in the first part of the results chapter are those suitable and applicable to visualize PRADS output data. The main criteria for selecting the visualization tools implemented in our prototype is the capability of the visualization tool to be automated. Another criteria for selecting the visualization tools is with regards to the file formats they take to visualize the log file. The output from PRADS is a CSV, comma separated value, file and the visualization tool, Afterglow, takes *CSV* files to visualize files in graphs while Graphviz takes the *.dot* files as an input to produce visual presentation.

As a result Afterglow and Graphviz are the two visualization tools implemented in our prototype. Afterglow can directly take the parsed CSV files from PRADS and produces graph language which will be the .dot file that will be again an input for the Graphviz. Finally different graphs ,according to the layouts of Graphviz chosen, will be presented for analysis and further inspection.These two visualization tools are easy to implement and suitable to consider the output data from PRADS. There is no involvement of data conversion which saves time and energy. The only thing to be considered is parsing the right fields from PRADS output and piping them to the Afterglow, whose output is again piped to the Graphviz. These combination creates a nearly real-time rendering

of graphs. In fact, Afterglow is used for both static and real-time log file visualization. However, Graphviz is used for static visualization systems.

The common behaviour for all the three visualization tools; Afterglow, Graphviz, and gephi is that all of them implement different customization mechanisms to present graphs in a meaningful and easy to look visual presentations. Afterglow and Graphviz have command line options to customize and filter their visualization process while gephi uses GUI options to do the same job.

Unlike Afterglow and Graphviz, gephi is run in a GUI manipulation mode where we specify the number of nodes, number of edges and type of graph. Gephi is not considered for our prototype implementation due to its dependency on GUI. During a sample test,gephi was observed to take input data manually and the customization process is very tiresome and time consuming. Gephi is commonly applied in visualization of social networking areas to show relationships between nodes. As a result we only considered the two previously discussed tools due to their dependency on each other and their ability to be easily automated. The advantage of gephi is that there is no limit on the input it takes for visualization. Gephi can visualize as many columns as possible with out any problem. There is no limit on the number of columns it considers for visualization and it is more interactive with users. Users can use their mouse to show mappings between nodes by hovering their mouse around the nodes and zooming in and out is done similarly in the GUI. But gephi is not recommended for sysadmins and it is not implemented here in the prototype.

However, PRADS logs its output in a CSV format and there is no need of changing a CSV log file to another CSV file format. The step required to visualize the logfile is to only pick out the fields that will be used for the visualization process and piping them to the Afterglow visualization tool,*Afterglow.pl*. Afterglow doesn't directly give us graphs, it gives us graph description files (.dot files) that will be further processed by another visualization tool such as Graphviz and is converted to graphs. Afterglow gives us many options to customize the output graphs we get from visualization. These options enable us to focus on the important parts of the data we want to visualize. The main functioning features that can be used in Afterglow for better visualization output are :

1.Filtering Nodes: In order to focus on the important parts of our network data, we can filter out nodes which do not fulfil a certain criteria. Nodes can be filtered based on the following criteria :

- Based on names : In the case of our prototype implementation for example TCP-flags were filtered out from the service fields in the visualization process. We can specify this filtering out process in the properties file of Afterglow.

- Based on number of occurrences : A minimum number of occurrences can be specified for a node to appear on the graph.
- Based on the number of connections a specific node makes with other nodes : Minimum number of connections can be specified as a limit for a node to be seen in the graph.

2.Colouring: Assigning different colours for the nodes in charge of the visualization gives better understanding of the graph. Colouring can be implemented to differentiate;

```
Nodes of similar behaviour
Edges of similar behaviour
```

3.Clustering : This is the most important and usable feature Afterglow gives us for better customization. We can cluster IPs of little interest for us and emphasize on the other IPs for example internal IP addresses, if we are interested to know our internal network topology. Clustering can be applied on nodes of similar trend in order to emphasize on the important nodes to be visualized.

Moreover Afterglow provides us with many command line argument options that enhance the visualization process.

Listing 5.1: Afterglow Command line Arguments

```
1  CSV file | perl Afterglow.pl -c properties [options] ...The main execution line.
2  -c properties-file is the most used option but there are other options like
3  -d -> to print the count of nodes
4  -e -> to assign edge length
5  -n -> to avoid label names
6  -o -> to give a limit on the number of nodes that should be shown on the visual graphs
7  -f -> to give a limit to the source node on how many connections the node should have to appear
8  on the graph
9  -g -> to give a limit for the event node on how many connections it should have to appear
10 on the graph
```

It was observed that these options really enhance the visualization process. In case of log file analysis with huge amount of data, it could be important for example to ignore those nodes with small number of connections. Moreover if one is interested in investigating ports scans, it is possible to show only the nodes or connections occurring more than a certain limit. When the target is also to know the internal network mapping details, the external nodes can be clustered to one node so that emphasis will be given on the internal network. Afterglow is explained here more than the other tools because it is the main tool implemented in our prototype and it is on this tool many customizations that determine the output graph contents are made. The drawback in Afterglow is that it only considers two or 3 columns of data for visualization;it doesn't accepts more than three fields for visualization. Although Afterglow is the main tool implemented in the prototype development, the output from Afterglow is textual graph language which needs further transformation as mentioned earlier

on this section.

Therefore Graphviz is also implemented in our prototype in conjunction with Afterglow to transform the textual graph languages from Afterglow output. The main contribution of Graphviz in the prototype is to specify the layout of graphs for better visual presentation. The graph layouts in Graphviz give different ways of presenting the data. Specifying the layouts gives us graphs either in two-dimensional or 3-dimensional presentations. The main layouts in Graphviz are :

Listing 5.2: Graphviz Layouts

```
1 dot : used when the number of nodes are less or customized to be less and used for hierarchical ↙
      ↳ layout
2 neato : for 3–Dimensional presentation, organic layout
3 circo : uses circular appraoch
4 twopi : uses radial layouts
```

The *dot* and *neato* are the layouts mainly implemented in our prototype. The dot layout is implemented when the number of nodes involved in the visualization are less while the neato layout is used to accommodate more number of nodes since it can tolerate node and edge congestion better than the dot layout. Similarly Graphviz is executed in conjunction with Afterglow as follows :

```
dotfile | [layout options]  -Tgif -o output.gif
where the dotfile is the output from Afterglow.
```

Part II

5.2 Suggested Visualization Methods

Different visualization ways have been suggested in this paper. The visualization ways described are all associated with PRADS output but those visualization ways mentioned are not all implemented in our prototype.

5.2.1 Bar Chart Visualization

The main goal of this thesis is the visualization of log files from PRADS, the visualization ways suggested using line charts and bar charts are not implemented in our prototype. These visualization methods are just suggestions and shouldn't be difficult to implement.This way of data presentation is merely to show the overview or general summary of the events in our network.

PRADS reports all the information presented in figure 4.3 in textual format also but it gives better analysis and impression when the information is visualized in line and bar charts . A user or network analyst can understand fast and easily about the active services and assets in their network by simply looking at the bar chart. The data used in this bar chart is obtained from the terminal line report found when PRADS ends its sniffing process properly. A user or network analyst can understand easily about the active services and assets in their network by simply looking at the chart.

The visualization shown in figure 4.4 represents the count of the client and server services being served in the network with in different time intervals of the day. It will report to the network analyst in which time of day more client is being served and in which interval more server-service is served. This can be implemented for anomaly detection in case there are suspicious activities in the network like port scanning or DNS service attack. when sysadmins have a daily report of the services running in their system, they can keep track of all the services and they will conclude if a certain service is down or not.

Similarly the bar chart visualization method shown in figure 4.5 shows the count of the ports participating in the network traffic at each time interval. This type of visualization can be used to show the access rate of ports at the corresponding time interval. This graph can create awareness among network and system administrators to be critical of the ports and associated services being accessed. Some attacks can be more at the time where the network and sysadmins are off and it is a good idea to have knowledge about the number of occurrence of the ports that have been active at a specific time. A network analyst can be suspicious when there is a unusual trend that a specific port is occurring more at a certain time interval. The network analyst can deduce if the situation is normal or not.

The visualization way presented in figure 4.6shows how the age of the services SSH and HTTP can be represented by different fading colours to show that they appear at different time intervals of a daily or weekly report generated by PRADS. This visual presentation is not limited to only those mentioned services it includes to all services seen in the network. The most important information for sysadmins is the current running services but it is also good to know the history of the services that were running at a time also. The fading colour might represent an older service or expired one. Such graphical presentation can be extended to all the services collected by PRADS at a certain time. This type of graphical presentation is possible because PRADS has options to show the new,ended and expired services based on the time stamp. PRADS can log the information it detects to database including the time stamp, as a result the age of each service can be known and shown as in figure 4.6. The idea behind the this figure's visual presentation is just to show the concept of applying fading colours to service observed at different time intervals because sysadmins are

interested at the current status of the clients and servers running in the network.

5.2.2 Tree Map Visualization

The visualization in figure 4.7 is one way of visualizing PRADS output data. It is a tree map visualization which shows the protocol-service mapping. This is implemented using the java based Afterglow version which makes use of the java library and is more interactive where the details of every node can be shown. The details for each service can be shown by zooming in and out the graph or clicking on each node. For example by clicking the SSH services, assets associated to SSH services will be displayed. This visualization way is more of a real-time visualization method which makes use of the java library and Afterglow 2.0 version. This visualization way is not implemented in this thesis with the prototype developed.

Figure 4.8 is similarly presented in tree maps. Clustering of similar assets is used in this presentation. This presentation is useful when we have large amount of data with so may IP addresses involved. Displaying all the IP addresses can create congestion and cluttering in the tree map, which makes clustering useful in this case. This way of representing IPs helps users and network analysts to have a general overview of their network. Clicking each subnet or zooming into each subnet displays the IP addresses of the subnet in charge. This presentation way is also presented using the java based Afterglow and it is not implemented in the prototype.

The visualization way represented in figure 4.9 is another tree map visualization showing the occurrence of the IP-port mapping in the log file. This type of visual presentation is helpful to identify some anomalies or suspicious activities. If a certain IP is connected to a specific port many times, then it could be a port scan or some sort of attack. By summarizing the appearance of a specific IP-port mapping , it can be summarized in different shapes and sizes as shown in the figure 4.9. This is also tree map visualization way which can be implemented using Afterglow 2.0. This node occurrence wise visualization can be implemented in our prototype using Afterglow 1.x version by configuring the properties file to calculate the occurrence of a node and change its size as demonstrated in figure 4.18 in part III of the results chapter.

5.2.3 Internal and External IP Mapping

Visualization based on internal and external IP mapping is one way of visualization suggested in the results chapter. PRADS log file doesn't show any

network traffic such as incoming and outgoing traffic. But PRADS has options that display the network traffic and connection tracking. The options are :

```
prads -O ...Connection tracking [O]utput - per-packet!
prads -x ...Conne[x]ion tracking output  - New, expired and ended
```

These options showing connections between IP addresses are shown in the terminal window; the connections are not logged to the log file by PRADS. The options show us the connection tracking per packet but PRADS still log its output in an eight-column field with the same parameters asset, vlan, port, protocol, service,[service-detail], distance, discovered. Therefore visualization mechanisms involving network traffic connections such as outgoing traffic and incoming traffic are not implemented in our prototype since PRADS logfile doesn't show such data. Consequently these two visualization ways described in figures 4.10 and 4.11 are not implemented in our prototype development.

Using these two options we can see the connections between the IPs communicating. Such visualization mechanisms can be presented using the hierarchical or link graph representations in figure 4.10. This method also implements clustering for the external IP addresses and shows internal IP and port mapping as well. In this case the external IPs can be clustered into one node with the purpose of network administrators prioritizing their internal network. We have to first know what is going on with our internal network before analysing external networks and external IPs. This figure shows how external IPs can be clustered to one node if the emphasis is going to be given for the internal IP addresses. The figure in general shows the incoming traffic where the source node is clustered into one node as External, and the event nodes are the internal IP addresses and the target nodes are the ports. The idea of source, event, and target nodes will be later implemented in the prototype.

The visualization mechanism shown in figure 4.11 maps the local IPs with their respective ports against the external IPs with their respective ports. Moreover the IP addresses will be running services with their associated ports and this leads to IP-Port-Service mapping. In the visualization mechanism presented in figure 4.11 the internal IPs are the servers giving different services at different ports. An example scenario could be considered using the HiOA network where external requests are coming in looking for services from the university college. On the other hand another visual presentation other than the one presented in figure 4.11 can be presented showing internal IPs of HiOA requesting services from the external world using global IPS. Such visualization is possible using the connection tracking options of PRADS but not from the log file yet. Therefore if PRADS is run with the right options that show traffic flow from source to destination, then implementing this visualization methods is a straight forward job. But since the goal of this thesis is visualizing PRADS log file to have better log analysis and network mapping, the prototype is developed with this perspective.

5.2.4 Visualization Using Asset-Port-Service Mapping

This section is directly associated with the prototype development, and the prototype implementation involves asset-port-service mapping since PRADS output log file contains these fields. Therefore the visualization mechanisms shown in figures 4.12,4.13 and 4.14 can be implemented in our prototype. The number of nodes involved in these figures are less for the sake of demonstration but the number of nodes can be more and there should be a way to accommodate more number of nodes using customization methods like clustering. The figures mentioned earlier in this subsection have different presentation ways even if they involve asset-port-service mapping.

The figure 4.12 shows how visualization is possible using asset and service mapping. In such a situation,one asset can be both client and server as shown in figure 4.12. In the figure the IP3 is supposed to represent external address where as IP1 and IP2 are internal IPs and different colouring is used to show the different nature of the nodes. The nodes that are shown in such graphical presentation can be filtered using different criteria like limiting the number of occurrences for a node to be shown in the graph and how many connections a node should have in order to appear in the graph to decrease graph congestion.

The visualization shown in figure 4.13 shows the IP-port-service mapping detected by PRADS in the network. In this case we have a clear distinction between source nodes which are the IP addresses involved, the event nodes which are the ports and finally the target nodes, services associated with the ports displayed in the event nodes. The target nodes are specified as server and clients. We call them services because they belong to the column services in the PRADS output format. This figure's representation is similar to the presentation in figure 4.12 except that there is some rearrangement in the nodes . Such visualization method is the one that is going to be implemented in the prototype development part. In this figure some nodes are acting as both server and client which is also true in the actual output of PRADS. As shown here IP3 can give webservices and at the same time it can run a web-browser acting like a client.

The other visualization method involving asset, port, and service mapping is shown in figure 4.14. This visualization mechanism is customized in a better way than the other two visualization ways mentioned in this subsection. The visualization way in figure 4.14 implements edge and node customizations. As the number of nodes and the size of the graph increases the complexity of the graph will also increase. At some point if unimportant nodes are not filtered out and if emphasis is not given only on the important nodes that we want them to appear in the graph, the graph will be saturated and we will arrive at a point where it is difficult to distinguish the asset-service mapping. As the increased number of mappings creates visual complexity, there should be a means to han-

dle such situations. In addition to the node customizations like clustering; one scenario that can decrease the complexity of mapping is edge-colouring. As the edges go from one node to another, as the number of nodes increases it could be difficult to identify which edge is coming from which node and which edge is going to which node. This situation is created as we have 3-level nodes called source,event and target, where the even node is the one that connects the source and target nodes. Therefore by colouring the edges, it will be easier to identify the asset-service mappings. It will be easier to show which host is acting as server at which port, and which asset is a client at which port. This visualization method is implemented and demonstrated in our prototype.

Part III

5.3 Demonstration of the Prototype

The prototype that automates the visualization process was tested and demonstrated in different scenarios by running PRADS to collect data from different networks. The visualization process is still the same in all scenarios but the difference is the amount of data collected by the sniffing tool, PRADS. When the internal network is considered for testing for instance, there won't be a lot of traffic going on since the traffic is controlled and generated deliberately for the sake of testing the prototype. But it is a different case when PRADS is listening on the HiOA network where there are many activities going on from inside and outside.

For the sake of simplicity and to have a clear idea of the prototype, it is worth discussing how the prototype functions. The prototype automates all the steps involved in visualization. The steps are data collection, parsing the data, visualization and presentation. The steps can be summarized using a perl script as follows.

Listing 5.3: Visualization Prototype Process

```
1  prads -i $interface -l $output_file
2  perl parser.pl -f $output_file | \
3  perl /home/group1/Afterglow/Afterglow-master/Afterglow.pl\
4  -c /home/group1/Afterglow/Afterglow-master/color.properties -v |\
5  dot -Tgif -o /home/group1/Afterglow/Afterglow-master/$output_afglow.gif
```

Data is first collected by specifying PRADS to listen on the selected interface and making PRADS to log its output to a file. Another parsing tool, parser.pl, is developed using a perl scripts that parses the right fields that we need for the visualization process. The parser pipes the parsed columns of the 8-fields PRADS output data and feeds it to the Afterglow visualization tool. Afterglow creates textual graph description files that are further visualized and presented

by Graphviz. Graphviz has different graph layout options that give us flexibility in our visualization. The main steps involved in visualization of the data collected by PRADS are the same but we get different visual presentation of the data according to the different customizations we make in the properties file of Afterglow and according to the layouts we use in Graphviz.

Therefore we have different results obtained by the prototype for the different scenarios specified. When PRADS was run in the internal interface eth1 of the network setup specified in figure 3.1 , the data collected was not huge and the visualization process can be implemented using both dot and neato layouts of Graphviz.It is also necessary to make customization on the properties file of Afterglow to visualize the nodes of bigger importance. It is possible to visualize everything detected by PRADS but in this case the most important parts of the network could be ignored which we don't want to happen. As a result we need to identify what we want to visualize and know about the network, so that we will have a useful visualization process. This implies every customization and specification we make with the properties file configuration is according to our point of interest.

Figure 4.15 gives emphasis on the common ports 22, 53, 80, 21 and 443 and are assigned different colours each and other ports are filtered out.This graph's visualization uses the **dot** layout since the number of nodes involved is small. To avoid confusion edge colouring is used here. The *service* field in the PRADS output contains TCP-flags such as SYN, FIN, RST , etc. But these TCP-flags are filtered out in our visualization since they are not considered important in our visualization demonstration. The customization and filtering for figure 4.15 is done in the properties file. The dot layout is used by Graphviz when the number of nodes to be visualized is small but it is possible to implement the neato layout as shown in figure 4.16. Clustering can also be applied to source nodes and event nodes as shown in figure 4.17. Such clustering can be made if we are not interested in one set of IPs and give more emphasis on the other set of IPs, internal IPs for instance.

The visualization shown in figure 4.18 is node customization based on the number of occurrences of the nodes. This is more interactive way of presentation which creates more impression for users and network analysts at the first glance. The size of a node increases when the number of occurrence increases. There is a fixed size of a node specified on the properties file and this fixed size will be multiplied by the number of occurrence of a node on the network. Here is where some anomalies or suspicious activities can be identified. If a certain IP is connected to a specific port many times, which will be shown as a larger node,then it could be a port scan or some sort of attack and network analysts will be obliged to analyze and investigate it.

As mentioned earlier, the visualization process of the data collected by making

PRADS listen to the interface eth0 of the network setup in figure 3.1 is done in similar way. The network data collected will be larger in this case which leads to the need of more customization and filtering of unimportant nodes. In addition, as the number of nodes is growing the dot layout will not be suitable to apply and the neato layout is preferred. But we can still use the dot layout by filtering out many of the unnecessary nodes. The visualization demonstrated in figure 4.19 implements the neato layout since the log file collected by PRADS is too large for the dot layout. Even though we are implementing the neato layout, it is a good idea to filter out the unimportant nodes like TCP-flags by specifying these nodes to be invisible in the properties file. If we filter these unimportant nodes in our visualization process, then it will be easier to concentrate on the most important nodes (fields) that can have more effect on our network.

If our purpose it to have the general idea of our internal network and not interested on the IPs of external network, we can still implement the dot layout of Graphviz by using clustering of nodes and filtering out of unimportant nodes. Using clustering and filtering, the visualization shown by figure 4.19 can be implemented in figure 4.21 which is more general.

Similarly data was collected by PRADS from the HiOA network where there is a lot of traffic. The data collected contains assets represented by both IPv4 and IPv6 addresses.For better visualization the assets are specified as internal assets and external assets in the properties file. The visualization process presented in figure 4.22 implements the neato layout since the data collected is too huge to be presented using the dot layout. This way of visual data presentation is used when emphasis is given to the common ports shown in figure 4.22. Such visualization helps sysadmins and network analysts to know which service or port is mostly accessed and makes them concentrate on the specific port with the more number of connections . Concentrating on the node with more number of connections, sysadmin can make further analysis and investigate whether there is abnormal traffic or not.

Sometimes specifying the common ports that we want them to be visualized in the graphs can create potential problem in our network system. We can be more aware of the attacks or vulnerabilities on the common ports and services running in our network. But there could be some services that are running on a network that sysadmins are not aware of. Usually, in network system,we are attacked not by the things that we see but by the things that we don't see. We commonly know that most service run in ports less than 1024, and there could be services that can potentially be attacked by attackers and create vulnerability to our network system. The visualization of the HiOA network data shown in figure 4.23 shows the importance of visualization that includes all the ports less than 1024 in the properties file and how visualization gives network analysts first impression at the first glance on the graph.Such visualization gives more eagerness for network and system administrators to know what these ports are.

Looking at the figure 4.23 we can be sure that additional services are running on those ports other than what we normally know about the common services such as SSH, DNS and HTTP services.

The visualization implemented in figure 4.24 shows how filtering out nodes works when automating the Afterglow in our prototype. It is possible to visualize only the source nodes that occurred more than a specific threshold. The visualization shown in figure 4.23 can be further reduced to the visualization shown in figure where a dot layout is implemented. When ever we see an IP connected more than the other IPs in charge of the visualization, we give more emphasis on that IP and tend to investigate it as sysadmins. Syadmins tend to filter nodes with a certain number of occurrence to study if an asset is deliberately trying to attack or if it is a normal connection. For instance if an IP is seen in the network more than 20 times, then emphasis should be given on this IP and this IP needs to be analyzed more.We can play more with the command line arguments of Afterglow to check the frequency of the occurrence of assets by specifying the arguments f and/or g.

Extended testing of the prototype in the open-stack, Alto demonstrated the same visualization mechanism is applied no matter how large or how small a network is. The visualization shown in figure 4.25 is implemented in the other scenarios demonstrated by the prototype. It can be observed from the visualization on the open-stack that the services running are limited to SSH-service and DNS-service and clients. This implies that the prototype can be implemented to visualize any asset and service which is detected by PRADS and the prototype can be used by network analysts to map their assets and services and know their respective client and server services. Similarly when the size of the data collected increases, it will be difficult to use the dot graph layout, the neato graph layout is rather used. This will result in a better visualization without congesting the nodes. The data collected was tested with the dot graph layout but the visual presentation was with cluttered nodes where it is difficult to see the asset-service mapping. But the visualization presented in figure 4.26 is the best visualization way for large data as has been explained in other similar scenarios. The data collected for graph 4.25 visualization was obtained by running PRADS for about 2 hours, but for the case of graph 4.26 the data was collected for about a week. The visualization in graph 4.26 shows one more service is detected by PRADS other than the DNS and SSH services shown in figure 4.25 and that is HTTP service and client.

Chapter 6

Discussion and Future Work

This chapter discusses the different parts of the project covered so far including the approach, results, and the process of the implementation. The chapter finally wraps up by suggesting improvements for future work and continuation of this project.

Part I

6.1 Evaluating PRADS and The Selected Open-source Visualization Tools

PRADS is a powerful asset detection tool which sniffs the network passively without generating any network traffic of its own. It is a very useful tool that network and system administrators should implement in their network. The main advantage network analysts can get from PRADS is that they will know about every service running in their system. Network analysts become aware of their system because PRADS reports everything that it sees in the network and log it to a log file. Analysing and investigating the log file collected from PRADS will help network and system administrators understand their network system. Analayzing and investigating the trends in PRADS text file is difficult and hard to interpret. As a result the importance of visualizing PRADS text file is considerable since visualization helps in tracking trends that are difficult to interpret from text files.

The open-source visualization tools Afterglow and Graphviz are selected for visualizing the contents of PRADS output data. The main reason these tools are selected and implemented as the visualization tools in this project is that the tools are better suited and usable to implement PRADS output data. Automating these two tools using the perl scripting language is easier when implementing

the prototype. Afterglow has perl scripts which produce graph descriptions from CSV files. Graphviz takes graph descriptions as input and using one of its layout features dot, neato, etc; produces visual presentations in the form of graphs or tree maps. The file formats the tools consider during the visualization process is also another advantage these tools give us over the other available tools. Afterglow and Graphviz depend on each other in the prototype implementation. Afterglow takes CSV files as its input and produces graph descriptions which are later used by Graphviz for visual presentations. In the prototype implementation, Afterglow is the first phase process of Graphviz because Graphviz depends on the output of Afterglow for further visualization. Consequently the performance of Afterglow will affect the fucntioning of Graphviz.

The idea of the first part of the problem statement was to to survey and find open-source visualization tools best suited to visualize the contents of PRADS output data. Afterglow is the best suited and most reliable tool for the visualization of PRADS output data as demonstrated by the prototype. The reason for this is that Afterglow contains perl scripts that are ideal to automate with other perl scripts to produce well customized graph descriptions where the graph descriptions will be further processed to give visual presentations . The drawback of Afterglow visualization tool is that it only considers two or three columns for visualization. It is not possible to have a mapping which relates more than three columns or fields. But to overcome such problems the visualization process can be made by grouping the fields by parts. Emphasis is given on the asset-service mapping, hence the visualization will consist of the asset, ports, and service. When visualization is required to map assets and the OS the hosts are running, this could be done separately and independently parsing the respective fields accordingly. In a similar project Afterglow and Graphviz were implemented to visualize Honeypot data where only the dot graph layout of Graphviz was implemented but not the neato graph layout which is implemented in case of this thesis.

Part II

6.2 Evaluating the Suggested Visualization Methods

A number of visualization methods for PRADS output data have been suggested in the results chapter. The suggested visualization ways are intended to help the implementation of the prototype in this project or to propose the visualization methods for future work if not possible in this project. In fact some of the visualization ways suggested helped in the development of the prototype. On the other hand some of the suggested visualization ways involving the bar charts and tree maps are not implemented using the prototype. Some suggested visualization ways involving the bar charts depend on the terminal window verbose

output obtained from PRADS, not on the log file, which makes it difficult to automate in the prototype. The visualization methods implemented in this thesis are those showing the asset-service mapping and asset-port mapping. Similarly the visualization methods consisting of source-nodes, event-nodes, and target-nodes are also implemented here.

Although many visualization ways are suggested for the implementation of PRADS log file visualization, those that are implemented visualization ways consist of asset, port and service fields of the PRADS output data. The proposed visualization ways benefit sysadmins by showing the mapping between asset and service. Sysadmins will know what servers and clients are running in their network system. Some of the visualization ways suggested are implemented in other visualization projects but they were customized with regards to PRADS output data. Especially the visualization ways consisting of network traffic flow, showing source IPs and destination IPs, are implemented in other log file visualization projects but they can still be implemented in PRADS data using the right argument options that show connection tracking. In general the proposed visualization methods are helpful to develop the prototype and to customize the execution process of the prototype.

Part III

6.3 Prototype Evaluation

The intention of the prototype implementation is to show the asset-service mapping of the network detected by the PRADS tool. The prototype incorporates the data collection, parsing of the fields, visualization and visual presentation. The prototype executes these processes in near real-time process, but not exactly real-time. There is some delay estimated to be in seconds when the data to be visualized is bigger since there are two pipe processes involved in the visualization process. But for visualizations involving smaller amount of data, the visualization process is very fast and graphs are rendered almost immediately. The visual presentation is in the form of graphs.

The results obtained after the execution of the prototype show that the implementation of the prototype went as planned.The graphs executed using the dot layout Graphviz implementation were similar to suggested visualization ways. When the neato graph layout option was applied in the prototype, it gave a three-dimensional view of the network mapping among assets, ports and the services they are running which was interesting. The result is very helpful to network analysts that the visual presentations provide an environment for the fast interpretation of patterns and other useful features. Moreover it was good achievement to see the details of graphs using zooming in and out. Therefore

network analysts can depend on such visual presentations obtained from the prototype execution to analyse and investigate their network rapidly. The results of the visualization process clearly represent the text files in the log file and the graphs represent what they were intended to show to the sysadmins. The graphs work according to the customizations and filtering made in the configuration file or properties file in the prototype.

A lot has been said on the importance of visualization to identify abnormal patterns and behaviours in our network. Many researchers discussed visualization with respect to intrusion detection systems and other security logs such as visual firewall log analysis and forensics. Some researchers used commercial visualization tools such as ArcSight for log visualization and correlation. Another work was made on "Visualisation of Honeypot Data Using Graphviz and Afterglow" where the visualization was made using the dot graph layout of Graphviz. The visualization principle of this paper on Honeypot visualization is similar to the one implemented in this project but the results are not the same due to the difference in the nature of the data being considered in both projects. No research was made on visualization of PRADS log file. This project is the first to deal with PRADS output data visualization and considering the helpful and powerful detection tool PRADS is, the project tries to make use of PRADS output data in the best possible way. The results obtained by executing the prototype are encouraging and satisfactory but they are only the indication to show that it is possible to have such visual presentations for the log files, and this is just the beginning, there is plenty of space for further improvements.

The prototype used to implement the visualization process is not a complicated one. It has been tested by collecting data from different scenarios and visualizing them. The prototype's flexibility to any type of data parsed in a CSV, comma separated value, file makes it useful for any human network analyst to implement it for their network mapping and analysis. Using the same data, if the prototype is executed hundred times or more, the same output will be retrieved. The prototype is flexible in a way that it is still possible to visualize other fields of the PRADS output data by only making some arrangements. Any one can use the prototype to visualize their network data collected by running PRADS; but before analysing the visual presentation they need to know about the visualization process such as what nodes need more emphasis, what nodes are the target nodes, source nodes, the event nodes, and other customization details. After the users of the prototype are briefed with the visualization process, then they can start using it. Any one will come up with the same results if they follow the steps to implement the prototype since everything is accessible for free starting from the data collection tool, PRADS, to the visualization tools. On the other hand, even if the prototype fulfils the requirements of the project, there could be better output in the case of real-time visualization of log files by improving the prototype.

6.4 General Evaluation of the Project

The main goal of the project is to find out a way to present the textual output data generated from PRADS in visual graphs. The implementation of this project mainly simplifies the work of network and system administrators by providing them a visual presentation of their network data to analyse and investigate it. The project tried to implement the network data visualization process starting from scratch by automating every step including the data collection, parsing, and the visual presentation. It can be concluded that the project fulfils the minimum requirements it was meant for, except that the real-time visualization part was not exactly applicable using the selected tools and the implemented prototype. But the execution of the data and rendering of the visual presentation is implemented to be near real-time.

The real-time visualization of PRADS output data could have been implemented using the java based version of Afterglow and using PRADS shared memory mode but due to some technical difficulties this was not possible. Not only technical difficulty but also there was not enough time allocated considering the newness of the project and the complexity in semantics and syntax of the PRADS data. Understanding how PRADS works and playing around with its output to relate and associate with the open-source visualization tools took awhile. Other alternative approaches mentioned could lead to the real-time visualization of PRADS data. The intention of the project was to visualize the respective textual log files and present the text lines in visual presentation either statically or in a real-time mode. Therefore the project has fulfilled what it was intended for in a near real-time mode. It can be further modified in a more robust and interactive way using similar or alternative approaches. But this is just the beginning and there is always a room for improvement.

There is a general approach followed to implement this project and other alternative approaches are also described in the approach chapter. There were three different approaches considered at the beginning. All the approaches mentioned have to deal with PRADS output data to give us visual presentations in the form of charts, tree maps, and graphs. The approach followed to address the research problem is the one with less complexity and involves a stepwise automation of the main features the project goes through. After selecting the relevant visualization tools for PRADS output data and setting up the experiment environment, data was collected by running PRADS on the selected network environment. This collected data contains eight-column data of different fields, which is not possible to visualize the data at the same time using the selected tools. Therefore another parsing tool is developed using perl scripting language. The parser parses the right fields for visualization and pipe them to the visualization tool. There are two visualization tools involved in the implementation of this project depending on each other. Afterglow takes the fields to be visualized from the parsing tool and creates graph descriptions

which further visualized using the Graphviz resulting in visual presentations in graphs. This is the approach followed to implement the whole project according to the problem statement specified.

Similar results could have been achieved using the other alternative approaches discussed in the methodology chapter.The approach which implements the idea of using databases is similar to the approach implemented using the prototype. To implement the database based approach PRADS needs to insert its output data to the database instead of putting it to a log file. Then MySQL commands will be used to select the required columns from the database and will be fed to the visualization tools for further visualization which will be similar to the one implemented in this project. The other alternative approach suggested for this project is little bit advanced and is used mainly for real-time visualization of the PRADS output. PRADS supports a ring buffer log, which can be used to write log directly to a visualising agent through shared memory where both PRADS and the visualization tool use the shared memory. This is possible because PRADS supports the -B option in order to log connections to ring buffer. This approach will need more technical ability and advanced research on how to simulate the visualization tools with PRADS through a shared memory. This cannot be implemented using the perl based Afterglow tool since the real-time allocation of nodes is very difficult to implement using the implemented prototype in an interactive way where we can see the nodes reallocating themselves in the graph. If the real-time visualization is to be implemented, newly detected assets or nodes will need to be reallocated in the graphs immediately which will be difficult to implement using the perl based tools selected. The java based Afterglow 2.0 version is used for real-time visualization and the shared memory based approach could be implemented using this tool. On the other hand we can say the real-time concept is fulfilled if we can tolerate seconds of visualization delay. In fact other real-time network monitoring tools like Munin update the visual graphs after some minutes estimated to be 5 minutes. If we see from this point of view the prototype implemented in this project can be said that it is implementing real-time visualization.

If the problem statement had been changed to only implement the visualization of PRADS output data in real-time, the approach would have been changed. The tools selected to implement the visualization process could have been changed as well. The approach would probably be the shared memory based approach since it gives better option for real-time visualization. The perl based Afterglow version would be implemented. The java based Afterglow version would have been a good choice in conjunction with the Graphviz. In fact it has been observed during the prototype implementation that the generation of visual graphs using the dot layout of Graphviz is a processor intensive job and doesn't tolerate big log file data. A delay was experienced while executing the prototype implemented in a dot layout longer than the neato layout of Graphviz. Therefore the java based version of Afterglow and the neato layout of Graphviz will be a good choice to implement in the case of real-time PRADS

log file visualization. Which implies that if the project was to be done again starting from scratch emphasising on the real-time visualization process, the database based and the shared memory based approaches would be better approaches to implement using the Afterglow/Graphviz engine.

6.5 Contribution of the Project

A lot has been said on the importance of PRADS to network and system administrators in other papers. PRADS reveals network and system analysts all the services running on the network including those service that they are not aware of, protecting the network from potential damages which could be caused due to unawareness of the services running. PRADS also shows the operating systems the hosts are running. PRADS reports everything it sees on the network. How do we access the data collected from this powerful tool in a better way for better network analysis and awareness is the question this project tries to answer. Human brain and eyes are attracted to visual presentations such as charts and graphs than text files. It is a natural tendency of humans to be more interested in attractive and interactive things than the normal things, it is undeniable fact of nature.The point is not to discuss how interesting visual things are for the eye but to emphasise how easy it is to recognize unusual patterns and trends from a visual presentation. This project tries to present the data collected by PRADS in a visual presentation to simplify the work load of network and system administrators. It simplifies the work of sysadmins in a way that looking at visual presentations is useful and easier in tracking or revealing properties that are challenging to interpret from text file presentations. The graphical nature of visual outputs creates an environment for the rapid interpretation of patterns and newly rising trends. The prototype helps network analysts to know the asset-service mapping in their network including the clients and servers available in their network.

It will be very difficult even for network analysts to interpret the graphs without knowing how the visualization was made. For the visual presentations to be interpreted in the right way, the network analyst should be familiar with the prototype, every customization and filtering made in the properties file of the visualization tool. In general, before the tool is used practically, users of the tool need to know about the syntax and working principle of the tool implemented in this project. If users understand the principle of the prototype they can customize the tool according to their requirements and point of interest. Otherwise the graphs created would only be interpreted by the creator of the graph if the users of the prototype do not know the working principle of the automated tool. Visualization of different log files is made with the idea of focusing on the very important issues that can affect our network. This project tries to focus on the big picture by emphasising on the important columns of PRADS data. The prototype implemented in this project is to visualize the

PRADS log file but it can be extended to visualize other security log files like the firewall log file. It only needs small modification to implement it for other log files which makes this project encouraging visualization to be used in every log file. The prototype implemented is not complicated and is easy to use, but there is still room for improvement. A lot could have been implemented depending on the idea of the project, but it is a new project and there was time constraint which made it difficult to implement everything. The rest will be suggested in the future works and is open for others to depend on.

6.6 Future Work

During the whole course of the research new problems were encountered and new questions started to appear, which creates an opportunity for further research. Many more things could have been implemented and tested, but was not due to time constraints. The author suggests further research work in the following domains:

- Real-time visualization of PRADS output data is not easy to implement; the visualization implemented so far is near real-time visualization, not more interactive in a way that we can see the new services and assets joining the network live. Thus more research needs to be done on how to visualize the log file and on how to select the right visualization tools for this specific purpose where we can see the new arrivals joining the visual presentation or graphs lively reallocating themselves as new nodes.

- The java based Afterglow version is used to make interactive graph and it will implement visualization in tree maps which is more interactive and real-time visualization process.
- The Afterglow/Graphviz engine is a processor intensive task and there is a delay during the visualization process especially when using the dot layout. More research needs to be made on how to tackle such computing problems during visualization.
- The tools used in this project for visualization do not provide us the possibility to write legends for the visual presentations so that anyone can understand the graph.
- The IP addresses of the assets detected by PRADS in the network are merely classified as internal IPs and external IPs, but there is no means to know or predict where the IPs detected on our network are coming from. It would be interesting if the geographical location of the IPs involved is known using IP to country conversions. After that the mapping could be made on world map. This could be achievable with more research and time allocation.

- Not all suggested visualization methods in the project are implemented. Those suggested visualization ways which involve clicking of nodes and placing the mouse pointer on the nodes to see details are left as future work.

Chapter 7

Summary and Conclusion

The problem statement for this thesis was put in three steps where the first step was to make a survey of open-source visualization tools that are suitable and applicable to visualize PRADS output data, the second part was to design and investigate different ways of visualizing PRADS output data using the available open-source visualization tools, and finally demonstrate a proof of concept to visualise statically and/or in real-time the host, service and connection data produced by PRADS. The problem statement requirements were answered and the implementation of the prototype was demonstrated in many scenarios of network environment and satisfactory results were obtained. While working with the project, additional research questions were raised and were suggested in the future works section. As a result the following tasks were completed answering the problem statement through the whole process of the project.

- A survey was made on the available open-source visualization tools investigating their usability and suitability to visualize PRADS output data. Afterglow and Graphviz were selected as the visualization tools to be implemented in the prototype because these tools were more suitable to visualize PRADS data and easier to automate.
- Different real-time and static Visualization methods for PRADS output data were investigated and evaluated; consequently possible visualization ways for PRADS data were proposed. Some of the proposed visualizations methods were implemented by the prototype developed.
- Finally a proof of concept was demonstrated by developing a prototype which incorporates the PRADS data and the selected open-source visualization tools. The proof of concept demonstration is done using a script developed to automate everything including the data collection, data parsing, data normalization, visualization, and presentation processes.

The project could have been approached differently especially for the real-time visualization case other than the one implemented. But this project implements

the visualization process near real-time, there is no much delay and it can be said the goal of the project was achieved. With larger organizations consisting of many assets in their network system, there could be even million of lines of log files which is very difficult to analyze and correlate the log files with all the assets unless the right method is implemented to analyse the log files . This is implemented by using visualization of log files, which this project has attempted to implement using different customization mechanisms. This project is a work in progress, nevertheless, there are some advantages a network analyst can get from using graphical approaches like this to make PRADS data analysis. The main goal of this project is to reduce the load of network analysts by helping them analyse their network rapidly using visual presentations. Visualization reduces the energy and time that would be spent in analysing many lines of log files since a graph is worth thousand lines of log files. Visualization makes the understanding of patterns and trends in the network data easier for network analysts.

PRADS is selected as the data source for this project because PRADS gives important benefits to network analysts by revealing the services running on the network which the sysadmin may not be aware of. These services can create potential vulnerability to the network system, hence PRADS lets the network analysts know that these services are running on their network . Visualization of PRADS data helps network and system analysts to have good understanding of the network mappings such as asset-service mapping. It gives a clear image of the clients and servers in the network. It will not be easy to find out at which port an asset is acting as a client and at which port it is acting as a server by simply looking at the log file of PRADS especially when we have huge amount of data collected. This is practically demonstrated using the prototype when data was collected from the HiOA network using PRADS, which was around 4500 lines of log file. It was not easy to look at the 4500 lines by scrolling down with the mouse but it was better and easier to analyse the log file using visualization, where the picture is analysed rapidly. The prototype developed in this project demonstrated that visualization simplifies the work of network and system administrators because the visualization process can be customized. The visualization process can not be customized to show everything but the most important ones according to our point of interest.

Bibliography

[1] António Alegria and André Vasconcelos. It architecture automatic verification: A network evidence-based approach. In *Research Challenges in Information Science (RCIS), 2010 Fourth International Conference on*, pages 1–12. IEEE, 2010.

[2] Cisco Visual Networking Index. Forecast and methodology, 2012–2017 cisco systems, usa, 2013.

[3] Daniel Huluka. Experiment using distributed high-interaction honeynet (d2h). 2013.

[4] Lingyu Wang, Sushil Jajodia, Anoop Singhal, Pengsu Cheng, and Steven Noel. k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities.

[5] N Wattanapongsakorn, E Wonghirunsombat, C Charnsripinyo, T Assawaniwed, V Hanchana, and S Srakaew. A network-based internet worm intrusion detection and prevention system. In *IT Convergence and Security (ICITCS), 2013 International Conference on*, pages 1–4. IEEE, 2013.

[6] De Montigny-Leboeuf, Frédéric Massicotte, et al. Passive network discovery for real time situation awareness. Technical report, DTIC Document, 2004.

[7] Petter Bjerke Falch. Investigating passive operating system detection. 2011.

[8] Sowmya CL, Mr CD Guruprakash, and M Siddappa. Visualization of network traffic.

[9] Sebastian Schmerl, Michael Vogel, René Rietz, and Hartmut Konig. Explorative visualization of log data to support forensic analysis and signature development. In *Systematic Approaches to Digital Forensic Engineering (SADFE), 2010 Fifth IEEE International Workshop on*, pages 109–118. IEEE, 2010.

[10] Davix Visualization visualization seminar. `http://www.youtube.com/watch?v=qgdShwrKl2w`. Accessed: 2014-04-02.

[11] Jon Oberheide, Michael Goff, and Manish Karir. Flamingo: Visualizing internet traffic. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 150–161. IEEE, 2006.

[12] Meghan Allen and Peter McLachlan. Nav network analysis visualization. *University of British Columbia,[Online, 29 May 2009]*, 2009.

[13] Huw Read, Andrew Blyth, and Iain Sutherland. A unified approach to network traffic and network security visualisation. In *Communications, 2009. ICC'09. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[14] Jostein Haukeli. False positive reduction through ids network awareness. 2012.

[15] PRADS-gamelinux prads-wiki. `https://github.com/gamelinux/prads/wiki`. Accessed: 2014-03-12.

[16] PRADS-man prads-documentation. `https://github.com/gamelinux/prads/blob/master/doc/prads.man`. Accessed: 2014-03-12.

[17] PRADS-Discription passive real-time asset detection system. `http://manpages.ubuntu.com/manpages/precise/man1/prads.1.html`. Accessed: 2014-03-12.

[18] flat file Databases flatfile db. `http://en.wikipedia.org/wiki/Flat_file_database`. Accessed: 2014-05-06.

[19] Michael Widenius and David Axmark. *MySQL reference manual: documentation from the source*. O'Reilly Media, Inc., 2002.

[20] Graeme Merrall. Php/mysql tutorial. *Webmonkey. com, January*, 2005.

[21] AB MySQL. Mysql, 2001.

[22] Josephsen David. Building a monitoring infrastructure with nagios, 2007.

[23] Greg Conti. *Security Data Visualization: Graphical Techniques for Network Analysis*. No Starch Press, 2007.

[24] DH-T Wong and Kok-Soon Chai. Statistical analysis learning approach: The use of artificial intelligence in network data visualization system. In *Computer Applications and Industrial Electronics (ICCAIE), 2010 International Conference on*, pages 206–210. IEEE, 2010.

[25] Craig Valli. Visualisation of honeypot data using graphviz and afterglow. *Journal of Digital Forensics, Security & Law*, 4(2), 2009.

[26] Adetokunbo Makanju, Stephen Brooks, A Nur Zincir-Heywood, and Evangelos E Milios. Logview: Visualizing event log clusters. In *Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on*, pages 99–108. IEEE, 2008.

[27] Security Visualization secvizual. `http://secviz.org/`. Accessed: 2014-04-01.

[28] John Ellson, Emden R Gansner, Eleftherios Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz and dynagraphstatic and dynamic graph drawing tools. In *Graph drawing software*, pages 127–148. Springer, 2004.

[29] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. In *ICWSM*, pages 361–362, 2009.

[30] gephi source gephi-source. `http://www.gephi.org/`. Accessed: 2014-03-20.

[31] Manuel Gomez Rodriguez. Mlss13: Network modeling and information propagation.

[32] NGO Gephi. Gephi-the open graph viz platform, 2010.

[33] cytoscape cytoscape-wiki. `http://en.wikipedia.org/wiki/Cytoscape`. Accessed: 2014-04-08.

[34] Michael E Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, and Trey Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, 2011.

[35] NodeXL nodexl-wiki. `http://en.wikipedia.org/wiki/NodeXL`. Accessed: 2014-04-09.

[36] Elizabeth M Bonsignore, Cody Dunne, Dana Rotman, Marc Smith, Tony Capone, Derek L Hansen, and Ben Shneiderman. First steps to netviz nirvana: evaluating social network analysis with nodexl. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 332–339. IEEE, 2009.

[37] Jeffery A Brown, AJ McGregor, and HW Braun. Network performance visualization: Insight through animation. In *PAM2000 Passive and Active Measurement Workshop, Apr*, pages 33–41, 2000.

[38] Doris Wong Hooi Ten, Selvakumar Manickam, Sureswaran Ramadass, and Hussein A Al Bazar. Study on advanced visualization tools in network monitoring platform. In *Computer Modeling and Simulation, 2009. EMS'09. Third UKSim European Symposium on*, pages 445–449. IEEE, 2009.

[39] vistools tools-wiki. `http://www.wikiviz.org/wiki/Tools`. Accessed: 2014-04-03.

[40] N Robison and J Scaparra. Interactive network active-traffic visualization. Technical report, Technical report, Texas A&M University, 2007. http://inav. scaparra. com/docs/whitePapers/INAV. pdf.

[41] vistoolsorg tools-org. `http://www.rumint.org/gregconti/publications/insecure_conti.pdf`. Accessed: 2014-04-03.

[42] Russ McRee. Security visualization: What you dont see can hurt you. *Information Systems Security Association (ISSA) Journal*, 2008.

[43] Visualizing Firewall Log Data to Detect Security Incidents trenton bond. http://www.giac.org/paper/gcia/1651/visualizing-firewall-log-data-detect-security/109883. Accessed: 2014-03-20.

[44] Mats Erik Klepsland. Passive asset detection using netflow. 2012.

[45] Raffael Marty. *Applied security visualization*. Addison-Wesley, 2009.

[46] Information Visualization visualization process. http://www.networkworld.com/community/node/41856. Accessed: 2014-05-02.

[47] afterglowviz secviz. http://afterglow.sourceforge.net/main.html. Accessed: 2014-03-10.

[48] afterglowproperties properties. https://github.com/zrlram/parsers. Accessed: 2014-05-01.

[49] afterglowfunctioning afterglowfunction. http://www.networkworld.com/community/node/41742. Accessed: 2014-05-01.

[50] graphviz secviz. http://www.graphviz.org/. Accessed: 2014-03-08.

[51] graphvizdiagram functioning. http://www.research.att.com/projects/IV_GRAPHVIZ/?fbid=TNCGRDFmuVm. Accessed: 2014-05-05.

[52] gephiviz secviz. https://gephi.org/. Accessed: 2014-03-08.

[53] Christopher P Lee, Jason Trost, Nicholas Gibbs, Raheem Beyah, and John A Copeland. Visual firewall: real-time network security monitor. In *Visualization for Computer Security, 2005.(VizSEC 05). IEEE Workshop on*, pages 129–136. IEEE, 2005.

Appendix A

Scripts

Listing A.1: parser.pl

```
#!/usr/bin/perl

use strict "vars";
use Getopt::Std;
use warnings;
use strict;
use DateTime;
#Global variables
my $VERBOSE = 0;
my $DEBUG = 0;

my $opt_string = "vdhf:";
getopts( "$opt_string", \my %opt) or usage() and exit(1);

$VERBOSE = 1 if $opt{'v'};
$DEBUG = 1 if $opt{'d'};

if ( $opt{'h'} ) {
    usage();
    exit 0;
    }

my $FILE = $opt{'f'};
debug ("File was $FILE\n");
( usage() and die "Please supply a filename\n" ) unless stat ($FILE);
open (LOG, "$FILE") or die "Error opening $FILE $!\n";
#my $iplist;
#my @lines = <LOG>;
#close(LOG);
#my @keys = split( /\s+/, $lines[0] );
#shift( @keys ); #to remove the # as the first field
#foreach my $line ( @lines ) {
# skipping if the line is empty or a comment
#next if ( $line=~/^\s*$/);
#next if ( $line=~/^\s*#/);
#my %hash;
#@hash{ @keys } = split( /\s+/,$line );
#while ( my $line = <LOG> ) {
 # if ( $line =~ s/.*?((\d{1,3}\.){3}\d{1,3}).*/$1/ ){

  # verbose("Detected IPs: $line\n");
    #$iplist = $line;
 # print "$iplist \n";
```

```
  # }
#}
#close(LOG);
#foreach my $line ( @lines ) {
# skipping if the line is empty or a comment
#next if ( $line=~/^\s*$/);
#next if ( $line=~/^\s*#/);
#my %hash;
#@hash{ @keys } = split( /\s+/,$line );
while (my $line = <LOG>) {
chomp $line;
#$line =~ s/\[([^\]]|(?0))*]//g;
#next if ( $line=~/[\]]/);
$line =~ s,\[.*?\],,; #Delete all text between square brackets
my @fields = split "," , $line;
#my @fields=grep { !/^[\[.*\]]$/ } split ",", $line;
my $sip = $fields[0];
my $dport = $fields[2];
my $service = $fields[4];
my $timestamp = $fields[7];
#my $timestamp = time;
my @months = ("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec");
my ($sec, $min, $hour, $day,$month,$year) = (localtime($timestamp))[0,1,2,3,4,5];
# You can use 'gmtime' for GMT/UTC dates instead of 'localtime'
#print "Unix time ".$time." converts to ".$months[$month]." ".$day.", ".($year+1900);
#print " ".$hour.":".$min.":".$sec."\n";
#push @parsed, [ $fields[0], $fields[2] ,$fields[4] ];
print "$sip,$dport,$service \n";
}
  close(LOG);
 sub usage {
    print "Usage:\n";
    print "-h for help\n";
    print "-v for verbose (more output)\n";
    print "-d for debug (even more)\n";
    print "-f <filename> for prads.log file\n";

    }

sub verbose {
        print "VERBOSE:" . $_[0] if ($VERBOSE or $DEBUG );
        }

sub debug {
        print "DEBUG: " . $_[0] if ($DEBUG );
        }
```

Listing A.2: visualizer.pl

```
#!/usr/bin/perl

  # Needed packages
   use Getopt::Std;
   use strict "vars";
  use warnings;
  use strict;

   # Global variables
  my $VERBOSE = 0;
  my $DEBUG = 0;

   ######################
  # handle flags and arguments
  # Example: c == "..c", c: == "..c argument"
 my $opt_string = "hvdf:";
 getopts( "$opt_string", \my %opt ) or usage() and exit 1;
```

```
18
19  # print help message if ..h is invoked
20  if ( $opt{'h'} ){
21  usage();
22  exit 0;
23  }
24
25  $VERBOSE = 1 if $opt{'v'};
26  $DEBUG = 1 if $opt{'d'};
27  my $dir = '.';
28  $dir = $opt{'d'} if $opt{'d'};
29  # main program content
30
31   my $interface = "eth0";
32   #my $separator = ",";
33   my $output_file = $opt{'f'};
34
35 # for (my $c=0; $c<=3; $c++){
36
37   $output_file = qx#date +%Y%m%d#;
38 #$output_file=$output_file.$c;
39   my $output_afglow = qx#date +%Y%m%d#;
40
41   chomp($output_file);
42   chomp($output_afglow);
43   $output_file.= "-prads.csv";
44   $output_afglow.= "-afterglow";
45   debug($output_file."\n");
46 system("prads -i $interface -l $output_file");
47
48 #sleep(20);
49 #system("killall -i prads");
50
51  system(" perl parser.pl -f $output_file | perl ↙
        ↪ /home/group1/afterglow/afterglow-master/afterglow.pl -c ↙
        ↪ /home/group1/afterglow/afterglow-master/color.properties -v | dot -Tgif -o ↙
        ↪ /home/group1/afterglow/afterglow-master/$output_afglow.gif");
52
53
54
55
56  ######################
57  # Helper routines
58
59
60  sub usage {
61  # prints the correct use of this script
62    print "Usage:\n";
63    print "-h Usage\n";
64    print "-v Verbose\n";
65    print "-d Debug\n";
66    }
67
68 sub verbose {
69  print $_[0] if ( $VERBOSE );
70  }
71
72  sub debug {
73 print $_[0] if ( $DEBUG );
74  }
```

Listing A.3: Verbose Output

```
1   perl parser.pl -f 20140505-prads.csv | perl ↙
        ↪ /home/group1/afterglow/afterglow-master/afterglow.pl \\
2 -c /home/group1/afterglow/afterglow-master/color.properties -v -n | dot -Tgif -o \\
```

```
/home/group1/afterglow-master/20140505-prads.csv.gif
Verbose mode is on.
Skipping 0 lines.
Reading a maximum of 999999 lines.
Split mode for events is 0.
Threshold 0.
Source Threshold 0.
Target Threshold 0.
Event Threshold 0.
Maximum Node Size 0.2.

---------- Property File:
---------- Done Reading Properties

Lines read so far: 1. Skipped: 0. Processed: 1====> Processing: 10.0.0.1 -> 58078 -> SYN
Lines read so far: 2. Skipped: 0. Processed: 2====> Processing: 10.0.0.2 -> 22 -> SYNACK
Lines read so far: 3. Skipped: 0. Processed: 3====> Processing: 10.0.0.2 -> 22 -> SERVER
Lines read so far: 4. Skipped: 0. Processed: 4====> Processing: 10.0.0.1 -> 22 -> CLIENT
Lines read so far: 5. Skipped: 0. Processed: 5====> Processing: 10.0.0.4 -> 22 -> SYNACK
Lines read so far: 6. Skipped: 0. Processed: 6====> Processing: 10.0.0.4 -> 22 -> SERVER
Lines read so far: 7. Skipped: 0. Processed: 7====> Processing: 10.0.1.2 -> 22 -> SYNACK
Lines read so far: 8. Skipped: 0. Processed: 8====> Processing: 10.0.1.2 -> 22 -> SERVER
Lines read so far: 9. Skipped: 0. Processed: 9====> Processing: 10.0.1.2 -> 53 -> CLIENT
Lines read so far: 10. Skipped: 0. Processed: 10====> Processing: 128.39.89.8 -> 53 -> ↙
    ↳ SERVER
Lines read so far: 11. Skipped: 0. Processed: 11====> Processing: 128.39.89.8 -> 53 -> ↙
    ↳ SERVER
Lines read so far: 12. Skipped: 0. Processed: 12====> Processing: 10.0.1.2 -> 46894 -> SYN
Lines read so far: 13. Skipped: 0. Processed: 13====> Processing: 173.252.110.27 -> 80 -> ↙
    ↳ SYNACK
Lines read so far: 14. Skipped: 0. Processed: 14====> Processing: 10.0.1.2 -> 80 -> CLIENT
Lines read so far: 15. Skipped: 0. Processed: 15====> Processing: 173.252.110.27 -> 80 -> ↙
    ↳ SERVER
Lines read so far: 16. Skipped: 0. Processed: 16====> Processing: 66.220.152.19 -> 80 -> ↙
    ↳ SYNACK
Lines read so far: 17. Skipped: 0. Processed: 17====> Processing: 66.220.152.19 -> 80 -> ↙
    ↳ SERVER
Lines read so far: 18. Skipped: 0. Processed: 18====> Processing: 10.0.1.2 -> 46894 -> FIN
Lines read so far: 19. Skipped: 0. Processed: 19====> Processing: 173.252.110.27 -> 80 -> FIN
Lines read so far: 20. Skipped: 0. Processed: 20====> Processing: 10.0.1.2 -> 443 -> CLIENT
Lines read so far: 21. Skipped: 0. Processed: 21====> Processing: 66.220.152.19 -> 443 -> ↙
    ↳ SERVER
Lines read so far: 22. Skipped: 0. Processed: 22====> Processing: 66.220.152.19 -> 80 -> FIN
Lines read so far: 23. Skipped: 0. Processed: 23====> Processing: 10.0.1.2 -> 60272 -> RST
Lines read so far: 24. Skipped: 0. Processed: 24====> Processing: 128.39.89.8 -> 53 -> ↙
    ↳ SERVER
```

Listing A.4: Afterglow Properties File

```
1
2  #color.source="blue" if ($fields[0]=~/^2001:700:700:..*/)
3  #color.source="blue" if ($fields[0]=~/^128\.39\..*/)
4  color.source="blue" if ($fields[0]=~/^10\.0\..*/)
5  color.source="red"
6  #cluster.source="External" if ($fields[0]!~/^128\.39\..*/)
7  #cluster.source="External" if ($fields[0]!~/^2001:700:700:..*/)
8  #cluster.source="External" if ($fields[0]!~/^10\.0\..*/)
9  #color.source="red" if (field() eq "External")
10 #color.source="blue"
11 color.event="yellow" if ($fields[1] eq "22")
12 color.event="lightblue" if ($fields[1] eq "53")
13 color.event="brown" if ($fields[1] eq "80")
14 color.event="lightyellow" if ($fields[1] eq "21")
15 color.event="magenta" if ($fields[1] eq "443")
16 #color.event="yellow" if ($fields[1]<1024)
17 color.event="invisible"
```

```
color.target="black" if ($fields[2]=~/^SERVER/)
color.target="green" if ($fields[2]=~/^CLIENT/)
color.target="invisible"
#color.target="yellow" if ($fields[1]<1024)
#color.target="lightblue"

#cluster.source="SERVER" if ($fields[2]="SERVER")
#color="black" if (field() eq "SERVER")
#cluster.source="CLIENT" if ($fields[1]="CLIENT")

#color="lightblue" if (field() eq "CLIENT")
#variable=@ports=qw(22 80 53 443);
#color="green" if (grep(/^\Q$fields[2]\E$/,@ports))
#color="blue"
#color.target="blue" if ($fields[2]<1024)
#color.target="lightblue"
#color.sourcetarget="pink"
color.edge="black" if ($fields[2]=~/^SERVER/)
color.edge="green" if ($fields[2]=~/^CLIENT/)
color.edge="invisible"
size.edge=1;
# Changing node labels:
#label=substr(field(),0,10)

# URL for nodes (used for graphviz to enable image map functionality)
# This is an exampel of how to use AfterGlow with Splunk
#url=http://localhost:8000/en-US/app/search/flashtimeline?q=%20\N%20starthoursago%3A%3A24

# Using node sizes:
#size.source=1;
#size.target=200
#maxNodeSize=0.2

# More complicated node size example:
# This will size the source nodes and event nodes based
# on the frequency of their occurrence. This is also a
# great example of how you can use internal AfterGlow variables
# in your configs.
#maxnodesize=1;
#size.source=$sourceCount{$sourceName};
#size.event=$eventCount{$eventName};
#size.event=4;
#size=2;
#sum.source=0;
#shape.target=triangle
```

Listing A.5: Prads Installation

```
sudo apt-get install git
git clone git://github.com/gamelinux/prads.git
cd prads/
ls -l
 make
apt-cache search pcap
sudo apt-get install libpcap-dev
make
sudo apt-get install build-essentials
sudo apt-get install build-essential
make
apt-cache search pcre | grep dev
sudo apt-get install libpcre3-dev
make
type prads
sudo make install
apt-cache search rst2man
sudo apt-cache search rst2man
```

```
19 sudo apt-cache search rst
20 sudo apt-cache search rst | grep man
21 sudo apt-cache search rst | grep man
22 sudo apt-cache search python-docutils
23 sudo apt-get install python-docutils
24 sudo make install
25 type prads
26 /usr/local/bin/prads -v
27 /usr/local/bin/prads -v -i eth1
28 sudo /usr/local/bin/prads -v -i eth1
```